UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Mohamed RAHALI »

« SDN and NFV networks: A new boost and opportunity for network tomography »

Thèse présentée et soutenue à Rennes, le « 17/12/2020 »
Unité de recherche :
Thèse N° : 2020/xxxx

Rapporteurs avant soutenance :
Hind CASTEL Professor, Telecom SudParis, Paris, France
Pedro CASAS chercheur senior , AIT (Austrian Institute of Technology), Vienne, Autriche

Composition du Jury :

Examinateurs :	Nancy PERROT	Ingénieur de Recherche, Orange-Labs, Paris, France
	César VIHO	Professeur, Université de Rennes 1, Rennes, France
Dir. de thèse :	Gerardo RUBINO	Directeur de recherche, INRIA, Rennes, France
Co-dir. de thèse :	Jean-Michel SANNER	Ingénieur de Recherche, Orange-Labs, Rennes, France

Invité(s) :
Yassine HADJAJ-AOUL Maître de conférences, Université de Rennes 1, Rennes, France

TABLE OF CONTENTS

RÉSUMÉ EN FRANÇAIS

Contexte

Le besoin en services de connectivité ne cesse pas de croître dans les télécommunications, avec des exigences de plus en plus grandes en termes de bande passante, de réduction de la latence et de nombre d'équipements connectés. L'évolution vers la nouvelle génération de réseaux mobiles 5G capables de répondre à ces attentes devient essentielle. La bande passante attendue pour les utilisateurs du réseau 5G est d'un ordre de grandeur plus grande que celle offerte par le réseau 4G (qui est de 10 Mbits/sec, en moyenne). En outre, le temps de réponse visé est de l'ordre de la msec pour certaines applications. Les nouveaux services proposés ont été classés en trois catégories [1] : ultra haut débit mobile, internet des objets massif et réseau ultra-fiable à très faible latence.

Ces nouveaux services, avec leurs exigences contradictoires, doivent partager la même infrastructure réseau. La nouvelle architecture 5G doit par ailleurs disposer des moyens nécessaires pour assurer leur cohabitation. La virtualisation jouera alors un rôle essentiel dans cette évolution. En effet, des techniques telles que les SDN (les réseaux mis en œuvre dans le logiciel) et les NFV (fonctions réseaux « virtuelles ») permettent d'ores et déjà la création rapide de réseaux virtuels appelés « slices » sur la même infrastructure physique tout en respectant les contraintes d'isolement et de qualité de service. Le concept de *slicing* permet de créer plusieurs niveaux d'abstraction de ressources et la prise en charge de plusieurs utilisateurs. Ces nouvelles fonctionnalités pourront offrir une grande souplesse dans la gestion du réseau. Cependant, la tâche de supervision devient beaucoup plus compliquée.

Dans un tel environnement, il est difficile de garantir une solution de supervision globale car chaque acteur aura son propre système avec des restrictions d'accès mutuelles permettant l'isolation du service. Il faut donc surmonter ces limitations tout en cherchant à maintenir une image précise de l'infrastructure. L'utilisation des mesures de bout en bout peut être une bonne solution dans ce cas. Ces données peuvent être facilement obtenues à partir du trafic collecté au périmètre du réseau sans avoir besoin d'autorisations d'accès supplémentaires pour aller voir ce qui se passe à l'intérieur ou de protocoles spécifiques. Ensuite, ces informations peuvent être analysées pour en déduire l'état interne du réseau sans effectuer des mesures directes. Cette approche est appelée tomographie des réseaux [2] [3] et elle a été bien étudiée dans la littérature. La plupart des cas d'usage traités portent sur des topologies classiques, mais ils peuvent être facilement adaptés à des nouvelles applications liées à la supervision des réseaux

virtuels.

La nouvelle architecture est conçue pour offrir non seulement une capacité et une disponibilité élevées, mais aussi des solutions automatiques et dynamiques pour l'orchestration de l'infrastructure et des services. L'objectif principal est de réduire l'intervention humaine et d'automatiser la gestion des services. Pour atteindre cet objectif, le système d'orchestration doit disposer d'une image précise de l'état des ressources et d'outils efficaces pour diagnostiquer les défaillances et prendre les mesures correctives. Ainsi, pour être en adéquation avec les progrès impressionnants promis par la 5G, le système d'orchestration devrait également se caractériser par un système de gestion des ressources doté d'une grande évolutivité et flexibilité. Donc, les futurs systèmes de supervision devront satisfaire aux critères principaux suivants :

- **Haute précision.** Pour assurer une gestion efficace du réseau, le système d'orchestration doit disposer d'une image précise des ressources disponibles et de la capacité à bien anticiper les demandes futures.
- **Flexibilité.** Les nouveaux outils de supervision doivent prendre en compte les particularités des nouvelles architectures adoptées par la 5G. La diversité des services supportés introduira des indicateurs de performance multiples et hétérogènes. La présence de plusieurs acteurs dans une infrastructure partagée introduit plus de complexité pour le système de supervision.
- **Programmabilité.** Grâce à la technologie SDN, les futurs réseaux seront caractérisés par une grande dynamique dans la mise en œuvre des ressources associée à une forte programmabilité. En effet, les outils de supervision devront avoir la capacité de déployer et d'adapter les stratégies utilisées en fonction de l'évolution de l'état du réseau.

Les objectifs de la supervision des réseaux sont multiples. La vérification de la configuration des équipements constitue, par exemple, une utilisation importante des outils de supervision. Ainsi, dans un environnement SDN/NFV, le nombre de configurations envoyées du contrôleur aux équipements augmentera puisque l'une des principales caractéristiques du SDN est la simplification de la configuration (maintenant centralisées) des équipements et du déploiement des applications réseaux. Pour garantir la fiabilité de l'opération de supervision, le choix de l'approche est crucial. Il existe une large gamme de méthodes qui diffèrent les unes des autres en termes de technologies utilisées, d'objectifs recherchés et de ressources consommées.

La gestion d'un grand réseau multi-domaines peut donc être une tâche extrêmement complexe en raison de toute cette complexité technique. Habituellement, les systèmes de supervision sont limités par les autorisations d'accès aux domaines administratifs concernés et par le manque de protocoles et des ressources ad hoc pour accéder aux ressources. En effet, des outils de supervision dédiés sont alors nécessaires pour mesurer l'état interne d'un domaine à partir de mesures externes. L'inférence des métriques internes du réseau à partir des mesures externes est désignée sous le terme de tomographie des réseaux [2] [4] [3]. Elle a été

bien étudiée dans la littérature. La plupart des cas d'usage traités portent sur des topologies classiques, mais ils peuvent être facilement adaptés à des nouvelles applications liées à la supervision des réseaux virtuels.

L'origine du terme "tomographie" vient du domaine du traitement de l'image, où il désigne une technique permettant de reconstruire les caractéristiques internes d'un objet à partir d'une série de mesures prises de l'extérieur. De manière similaire, pour les réseaux de communication la tomographie vise à construire une image complète de l'état du réseau interne à partir d'une vue partielle élaborée à partir de mesures externes. Il s'agit d'un domaine de recherche actif depuis deux décennies. L'un des avantages de cette technique est qu'elles ne nécessitent pas d'accès supplémentaires pour les équipements ni de protocoles spécifiques pour collecter des statistiques. Les mesures de bout en bout peuvent être collectées soit passivement en échantillonnant les flux existants, soit activement en injectant un trafic spécifique qui peut être acheminé comme le trafic d'un utilisateur régulier.

Les premières solutions proposées en tomographie se concentraient sur l'inférence des paramètres internes (métriques des liens ou des nœuds) à partir de mesures externes. En effet, les chemins parcourus par les données dans le réseau et la stratégie de supervision sont souvent prédéfinis et l'objectif consiste alors à dériver la meilleure estimation des variables inconnues à partir des informations disponibles. Les mesures des liens/nœuds sont alors modélisées comme des variables aléatoires. Diverses approches statistiques peuvent être mises en œuvre pour estimer les paramètres de leurs distributions. Les estimateurs proposés varient en fonction des propriétés de la méthode de sondage appliquée et des protocoles utilisés (Unicast/Multicast, TCP/UDP, etc.). Des techniques statistiques diverses peuvent être appliquées pour approximer les paramètres inconnus tels que les estimateurs de maximum de vraisemblance avec l'algorithme de maximisation des espérances (EM) [5] [6], ou les méthodes des moindres carrés et les estimateurs bayésiens [7].

Récemment, une plus grande attention a été accordée à la conception des chemins de sondage et aux critères topologiques qui permettent une estimation précise des paramètres internes de ces distributions. Dans [8], les auteurs proposent une méthode statistique pour la sélection des chemins de sondage. D'autres approches possibles sont de nature purement algébrique. Dans ces méthodes, les métriques de lien/nœud sont considérées comme des constantes inconnues. Les mesures de bout en bout représentent des valeurs déterministes. Les solutions proposées se concentrent sur les conditions nécessaires sur les trajets de sondage permettant de déterminer uniquement d'une manière unique les paramètres de lien/nœoeud. Par exemple, pour les mesures additives telles que le retard ou la gigue, le nombre de chemins parcourus doit être égal au nombre de variables inconnues et ils doivent être linéairement indépendants [9]. En général, ces contraintes exprimées en termes de conditions algébriques sont souvent difficiles à exploiter dans la conception des stratégies de sur-

veillance. C'est pourquoi nous essayons on essaye toujours de les transformer en termes de propriétés de graphes, qui sont plus faciles à interpréter et à manipuler dans les problèmes de réseau. La majorité de ces solutions se concentre sur les métriques additives [9] et booléennes [10], où le problème inverse consiste à résoudre un système d'équations linéaires ou booléennes respectivement. Ces méthodes garantissent l'identifiabilité de toutes les métriques inconnues, mais plusieurs conditions et contraintes sur les points de collecte et les chemins de sondage doivent être respectées.

Contribution 1

Algorithme d'échantillonnage évolutif

Dans notre première contribution, nous avons étudié l'inférence de métriques additives sur des liens à partir des mesures unicast de bout en bout. Fixons quelques notations : nous avons L liens sur lesquels on cherche à évaluer la métrique, et P chemins sur lesquels on a la mesure de bout en bout. Soit A la matrice de P lignes et L colonnes où $A_{p,\ell} = 1$ ssi le $p$ème chemin passe par le ℓème lien, et $= 0$ sinon. Si X_ℓ est la valeur de la métrique sur ℓ et Y_p sa valeur sur le $p$ème chemin, et si X (resp. Y) est le vecteur des X_ℓ (resp. des Y_p), l'additivité signifie que $AX = Y$. Le problème est donc celui d'évaluer X connaissant Y, sachant que l'on a $P < L$.

Tout d'abord, nous étendons l'applicabilité d'une solution basée sur l'algorithme EM pour les topologies en arbre à des topologies arbitraires. Cependant, cette approche ne peut être appliquée qu'avec des graphes de petites dimensions car elle est coûteuse en temps de calcul. Pour cette raison, nous proposons ESA (Evolutionary Sampling Algorithm, c'est-à-dire, Algorithme d'échantillonnage évolutif), qui améliore considérablement les performances des algorithmes précédemment proposés basés sur la procédure EM. Le processus général s'inspire des étapes de l'algorithme EM, et les étapes élémentaires sont basées sur les principes des algorithmes génétiques dans lesquels nous essayons d'améliorer d'une manière itérative une population de solutions tirée au hasard.

ESA se base sur un processus stochastique. Globalement, une population de solutions est choisie aléatoirement pour vérifier une certaine contrainte. L'objectif est d'améliorer peu à peu la population pour trouver la meilleure solution au fil des itérations. Ainsi, au lieu d'attaquer le système d'équations linéaires $AX = Y$ comme dans les solutions basées dans la méthode EM, l'équation de référence est transformée en une inéquation en utilisant une borne ε de l'erreur commise à minimiser comme indiqué dans l'inégalité (1) :

$$||AX - Y|| < \varepsilon. \tag{1}$$

In (1), $||\cdot|||$ désigne une norme matricielle standard, X est le vecteur aléatoire qui représente les métriques inconnues, α est sa distribution de probabilité et Y est le vecteur des mesures réalisées de bout en bout. Comme nous le verrons, l'objectif est de trouver la meilleure distribution α avec le plus petit ε possible, au fil des itérations. La distribution α est initialisée de façon aléatoire. La première étape de l'algorithme consiste à rechercher une meilleure distribution satisfaisant l'inégalité (1) en procédant comme suit. Premièrement, un nombre important d'échantillons de vecteurs solutions (la population initiale) sont générés en fonction de la densité de probabilité initiale (par exemple, uniforme), dénotée $\mathcal{X}^{rand}$. Ensuite, les échantillons qui satisfont l'inégalité (1) sont sélectionnés ; $\mathcal{X}^{\text{selected}}$ désigne les échantillons sélectionnés. La nouvelle distribution de probabilité α est calculée à partir de $\mathcal{X}^{\text{selected}}$. Le pourcentage des échantillons sélectionnés $\mathcal{X}^{\text{selected}}$ doit être supérieur à un seuil minimum fixé, noté D, pour assurer la diversité de la population sélectionnée. Dans le cas contraire, l'algorithme convergera rapidement vers un optimum local. Dans les tests, nous fixons empiriquement D à 5%. La probabilité d'observer la valeur j sur le lien ℓ, $\alpha(\ell, j)$, est calculée en divisant le nombre d'échantillons où la métrique du lien ℓ prend valeur $j\Delta$ par le nombre total d'échantillons sélectionnés $|\mathcal{X}^{\text{selected}}|$, comme décrit dans (2), où Δ désigne le pas d'échantionnage.

$$\alpha(\ell, j) = \frac{|x \in \mathcal{X}^{selected}, x[\ell] = j\Delta|}{|\mathcal{X}^{selected}|}. \tag{2}$$

Ce processus est répété jusqu'à la convergence de α, ou jusqu'à atteindre un nombre maximum fixé d'itérations. Nous considérons que la densité de probabilité α a convergé si certains critères standard comparant deux instances consécutives (ou séparées par un nombre fixé d'itérations) de α sont satisfaits, comme dans de nombreux processus itératifs. Une fois que la distribution α est juguée avoir convergé, on change ε par une valeur plus petite et on recommence.

L'algorithme ESA comprend donc deux parties dont la complexité de calcul est pénalisante par rapport à l'ensemble de l'algorithme. Cette complexité est principalement déterminée par le nombre de solutions générées $|\mathcal{X}^{\text{rand}}|$. L'opération la plus coûteuse est la multiplication de la matrice des solutions aléatoires $\mathcal{X}^{\text{rand}}$ par la matrice A. La complexité de cette opération est en $O(|\mathcal{X}^{\text{rand}}|LP)$, où L et P désignent le nombre de liens et de chemins respectivement. La complexité des autres opérations, comme la génération de $\mathcal{X}^{\text{rand}}$ et le calcul de la nouvelle valeur de α avec (2), dépend de $|\mathcal{X}^{\text{rand}}|$ et de la granularité J. Leur complexité peut être approchée par $O(|\mathcal{X}^{\text{rand}}|J)$. Ces opérations sont répétées jusqu'à la convergence de α. On note n_1 le nombre d'itérations pour la convergence de α. Les deux étapes sont en fait répétées jusqu'à ce que la valeur ε soit considérée comme ayant convergé à son tour. Notons par n_2 le nombre d'itérations globales nécessaires. Les deux valeurs n_1 et n_2 sont principalement influencées par les paramètres initialisés comme ε et les conditions de convergence de α. La complexité

globale de ESA peut alors être approchée par

$$O\Big(|\mathcal{X}^{\text{rand}}|(LP+J)n_1n_2\Big).$$

Algorithme d'échantillonnage évolutif rapide

Nous proposons ensuite l'algorithme F-ESA (Fast ESA) qui réduit encore le temps de calcul tout en conservant une bonne précision par rapport à l'approche ESA. F-ESA est le résultat d'une analyse fine de l'impact des différents paramètres d'ESA sur les performances de la méthode. Les Figures 1 et 2 présentent respectivement le pourcentage d'erreur et le temps de calcul moyen pour les tests effectués afin de comparer les deux approches.

En ce qui concerne le pourcentage d'erreur, les deux algorithmes ont généralement les mêmes performances. La différence entre eux ne dépasse pas 1%, sauf dans un petit nombre de cas, comme pour J égal à 50, où l'erreur avec l'algorithme ESA est inférieure de 2% à celle de F-ESA. La différence entre eux est significative en ce qui concerne le temps de calcul. La Figure 2 montre que les temps d'exécution augmentent beaucoup plus vite avec la solution ESA lorsque J augmente qu'avec F-ESA, tout en gardant approximativement le même taux d'erreur. Par exemple, avec J égal à 60, le temps de calcul est d'environ 26 secondes avec ESA, alors qu'il est inférieur à 9 secondes avec F-ESA.

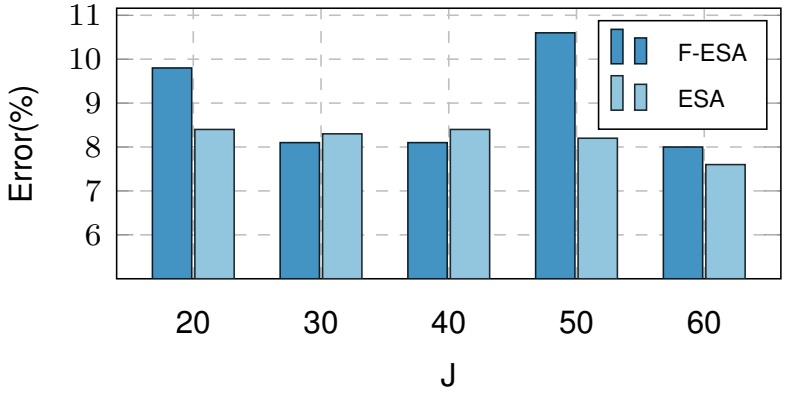


FIGURE 1 – Comparaison entre le pourcentage d'erreur pour F-ESA et ESA pour différentes valeurs de J (Correspond à la Figure 3.19 dans le manuscrit).

Nous abordons brièvement ensuite l'utilisation de l'approche ESA dans le cas de mesures non additives telle que la bande passante disponible, pour illustrer que ESA (et bien entendu, F-ESA) ne se limitent pas aux métriques additives. Pour ces dernières, le processus f qui donne la mesure de bout en bout à partir de celles sur les liens est linéaire : $f(X) = AX = Y$. L'étape d'inférence correspond ensuite au problème inverse associé. Pour les métriques non additives, nous avons toujours un problème inverse, seule la forme de f change. Les spécificités de

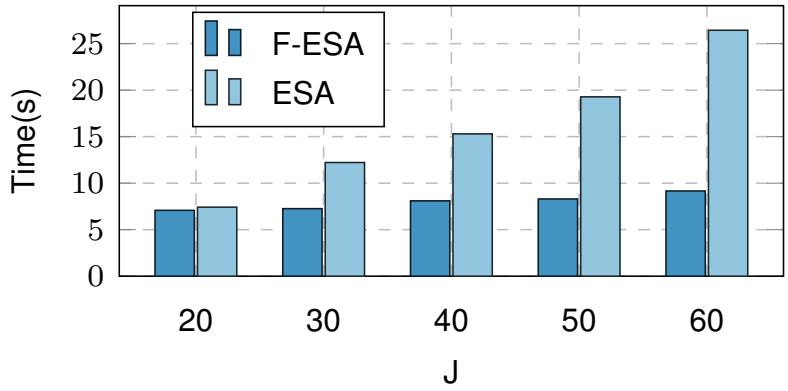


FIGURE 2 – Comparaison entre le temps de calcul pour F-ESA et ESA pour différentes valeurs de J (Correspond à la Figure 3.20 dans le manuscrit).

chaque métrique nécessiterons alors une adaptation du processus d'optimisation, mais les principes des algorithmes restent valides.

Un exemple intéressant de mesure non additive est la valeur de la bande passante disponible sur un chemin, qui correspond à la bande passante minimale disponible sur les liens qui composent le chemin. Ainsi, chaque métrique de chemin $Y(i)$ peut être calculée par (3) :

$$Y(i) = \min_{j \in [1,V] \& A(i,j)=1} X(j) \tag{3}$$

L'objectif de la tâche d'inférence dans ce cas est d'estimer la limite minimale la plus proche de la valeur réelle de chaque métrique de lien. Cela permet d'identifier les liens de congestion par exemple. Des méthodes sophistiquées sont disponibles pour mesurer la bande passante de bout en bout avec une bonne précision et sans saturer le chemin [11]. Donc, la procédure suivie par la technique ESA peut être facilement adaptée au cas non linéaire.

Contribution 2

La deuxième contribution traite le même problème d'inférence mais en suivant une approche d'apprentissage automatique. Nous considérons une architecture classique de réseau de neurones du type perceptron multicouches pour apprendre la relation entre les métriques des liens et celle des chemins. Le modèle apprend uniquement avec des données simulées et donne des résultats assez précis, très proches des résultats de notre méthode statistique ESA. La solution nécessite un petit volume de données ce qui rend le temps d'apprentissage très rapide. La caractéristique principale de l'apprentissage supervisé est sa capacité de trouver des relations complexes entre les entrées et les sorties correspondantes. De nombreux modèles efficaces ont été proposés dans la littérature pour cette tâche. Nous proposons de modéliser

notre problème comme une régression avec l'architecture de réseau de neurones précédemment mentionnée, en utilisant seulement 3 couches.

La dimension de la couche d'entrée est égale au nombre de chemins P, tandis que le nombre L de liens est le nombre de sorties. La dimension de la couche cachée est un paramètre variable. Dans notre proposition, nous n'utilisons qu'une seule couche cachée, ce qui s'est montré suffisant dans nos tests pour obtenir des résultats précis. Cette couche doit être d'une dimension suffisamment grande par rapport à celles des couches d'entrée et de sortie. Ainsi, elle pourra offrir de multiples combinaisons possibles pour décrire les sorties comme combinaison des entrées. À un moment donné, l'augmentation de la dimension des couches intermédiaires n'a aucun effet sur la réduction de l'erreur entre les valeurs réelles et les estimations. Dans notre mise en œuvre, nous avons choisi la fonction d'activation ReLu car elle offre un bon compromis entre coût de calcul et précision obtenue dans l'apprentissage.

La Figure 3 illustre les résultats des des tests déjà effectués avec les différentes solutions : SVD, k-paths, ESA et NN (réseau de neurones). Pour les tests effectués avec k-paths et ESA, nous calculons l'espérance des mesures sur les liens à partir des distributions de probabilité trouvées. La technique k-paths est testée uniquement avec une granularité égale à 4 et ESA avec deux valeurs de granularité, 4 et 50. L'erreur avec SDV et k-paths est très élevée car ces solutions ne conviennent pas pour des topologies relativement grandes. Les solutions ESA-50 et NN donnent généralement de bons résultats et les précisions sont très proches. Cependant, la méthode d'apprentissage automatique est plus rapide car l'étape d'apprentissage n'est déroulée qu'une seule fois. Ensuite, l'étape d'inférence est très rapide.

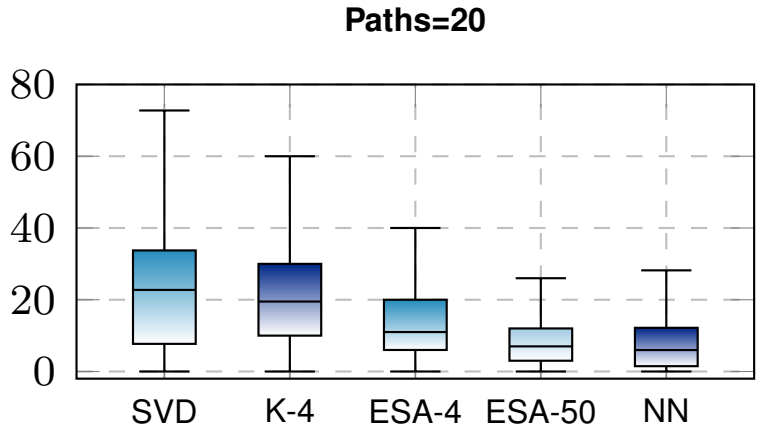


FIGURE 3 – Comparaison entre les différentes solutions proposées. (Correspond à la Figure 4.10 dans le manuscrit)

Contribution 3

Dans la troisième contribution, nous avons étudié le problème du placement des moniteurs, qui est complémentaire à la tâche d'inférence. Du côté des métriques, nous avons abordé ici le problème de la détection des liens défaillants, à partir de l'observation de certains des chemins choisis qui présentent des défaillances.

Les contraintes de délai dans les nouveaux réseaux est de l'ordre de la milliseconde. Les systèmes de supervision doivent être très précis pour effectuer des mesures à cette échelle. Donc, les moniteurs doivent être hautement synchronisés, ce qui n'est pas facile dans des conditions réalistes. C'est pourquoi nous avons proposé d'utiliser un modèle composé de cycles connectées aux moniteurs/sondes garantissant une synchronisation entre la source et la destination. Le problème est modélisé comme une tache de recouvrement d'ensembles et nous le résolvons en utilisant deux méthodes, un algorithme optimal et une approche heuristique.

La couverture de tous les liens du réseau par les cycles de sondage attachés aux moniteurs/sondes doit pouvoir permettre de détecter toutes les anomalies. Par exemple, si le délai sur un lien augmente à cause d'une congestion, cela introduit une augmentation du délai de bout en bout pour tous les cycles qui couvrent ce lien. A partir de ce constat, notre première stratégie de placement des moniteurs, appelée LinkCovering, a simplement pour objectif de couvrir le nombre maximum de liens sans introduire de contrainte supplémentaire.

La tâche de placement des moniteurs peut être formulée comme un problème d'optimisation où l'objectif est de minimiser le nombre de moniteurs déployés tout en satisfaisant les meilleurs critères de couverture de la topologie. Cependant, si une défaillance est détectée, il est difficile de la localiser exactement. Cela nécessite de calculer le vecteur des métriques X. Donc, la matrice A doit être carrée et non singulière. Cette contrainte peut être difficile ou impossible à satisfaire dans des conditions réalistes. Une stratégie possible consiste alors à rechercher le maximum d'équations linéairement indépendantes pour calculer une bonne approximation de X. Ceci nous amène à notre deuxième stratégie pour le placement des moniteurs, appelée MaxRank, dont le but est de choisir les cycles qui maximisent le rang de A.

Nous réalisons un prototype et évaluons la solution avec l'émulateur Mininet et une mise en œuvre réalisée avec le langage P4 du routage à la source pour construire les cycles connectés aux moniteurs/sondes. Pour chaque expérience, nous choisissons au hasard un lien pour générer des anomalies et l'algorithme essaye de les localiser.

Les moniteurs sont implémentés sous la forme d'un code `Python` qui est exécuté sur des sondes connectés aux commutateurs. Le module "Cycle Probing" sélectionne les placements des moniteurs et les déploie sur la topologie du réseau. Chaque moniteur construit les paquets à l'aide de la bibliothèque `Scapy Python` [12]. Le chemin de transmission est empilé

dans l'entête du paquet. Le nombre de paquets envoyées à chaque chemin est un paramètre configurable. Chaque paquet empile dans l'entête le chemin d'acheminement pour suivre le cycle correspondant et retourner à la au point de départ qui enregistre le temps d'envoi et de retour avec l'identifiant de paquet associé. Le moniteur calcule les délais de bout en bout et envoie l'information au module "Anomaly Detection". Cette entité agrège les mesures de tous les moniteurs déployés et les croise pour analyser l'état du réseau et localiser les anomalies potentielles à l'aide de l'algorithme ESA. Le module "Détection d'anomalies" calcule la probabilité de défaillance sur chaque lien. Ensuite, une alarme peut être activée si cette probabilité dépasse un seuil fixe noté p^*. Par conséquent, cette tâche peut être considérée comme une classification binaire, et les résultats pour chaque seuil fixé peuvent être résumés par quatre mesures qui sont décrites dans le tableau 1.

TABLE 1 – Paramètres de classification binaire

Variable	**Description**
Vrai positifs (TP)	Lien défaillant détecté
Faux positifs (FP)	Faux avertissement
Vrai négatifs (TN)	Bonne prédiction des liens normaux
Faux négatifs (FN)	Anomalie non détéctée

En général, la plupart des échantillons négatifs (liens normaux) sont faciles à prévoir, en particulier, les liens qui sont loin de la zone de congestion du réseau. Par conséquent, le nombre de vrais négatifs est toujours élevé, ce qui conduit à un indicateur qui ne permet pas d'évaluer correctement la performance de la classification binaire. C'est un cas typique auquel on est généralement confronté lorsqu'on traite des classes très déséquilibrées [13].

Les mesures de "précision" (Precision) et de "rappel" (Recall) sont bien adaptées pour évaluer un modèle de classification binaire avec des données déséquilibrées [14]. Ces métriques se concentrent davantage sur la classe positive et ne sont pas affectées par le grand nombre de vrais négatifs comme décrit dans les formules (4).

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}. \tag{4}$$

La Figure 4 montre la dispersion des probabilités sur les tests effectués. Ces valeurs sont utilisées pour tracer la courbe présentée dans la Figure 5. Nous utilisons le score AUC (Area Under the Curve) comme indicateur global pour évaluer les performances : plus l'aire est élevée, meilleure est la précision du modèle. Nous effectuons des tests similaires avec les deux stratégies de sondage. La stratégie de sondage MaxRank est plus performante que la stratégie LinkCovering. La AUC de MaxRank est de $0,8$, alors qu'elle est de $0,67$ pour LinkCovering.

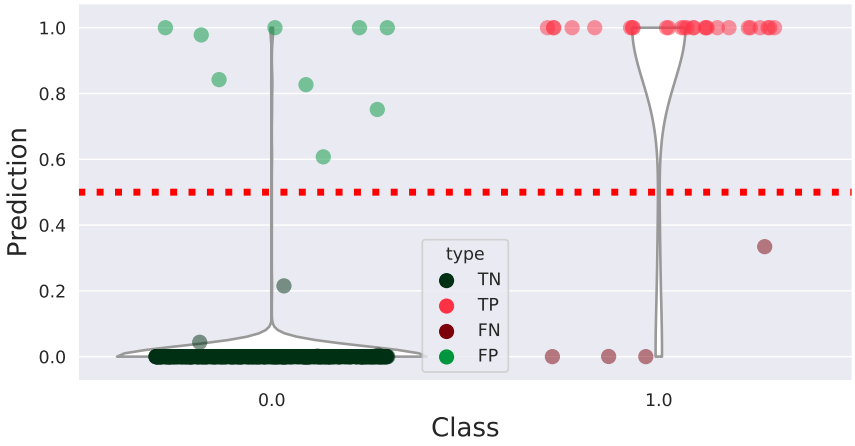


FIGURE 4 – Distributions de probabilité des classes (Correspond à la Figure 5.8 dans le manuscrit).

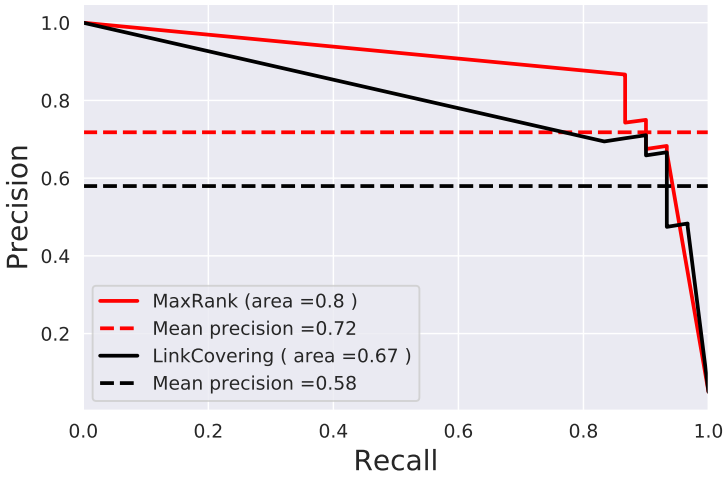


FIGURE 5 – Courbe de précision et de rappel (Correspond à la Figure 5.9 dans le manuscrit).

Contribution 4

Pour la dernière contribution, nous avons étudié la localisation des nœuds défaillants dans les réseaux NFV. Nous considérons une métrique booléenne où chaque nœud ne peut avoir que deux états, soit opérationnel, soit défaillant. Tout d'abord, nous donnons des conditions nécessaires et suffisantes pour la localisation de k défaillances simultanées dans une infrastructure virtuelle. Ensuite, sur la base de ces résultats, nous proposons un algorithme pour concevoir la topologie du réseau qui respecte ces conditions. Le problème est ensuite formulé comme un algorithme de correspondance et résolu selon un algorithme de maximisation de flot. La solution est testée à l'aide de simulations et les résultats sont conformes aux prévisions théoriques.

Notre solution permet de concevoir une stratégie de sondage permettant de localiser avec précision un maximum de nœuds défaillants. Pour évaluer la robustesse des topologies, nous procédons, en premier lieu, en générant de manière aléatoire le graphe correspondant. Ensuite, l'algorithme de correspondance est appliqué pour sélectionner un nombre minimum de liens supervisés permettant de détecter jusqu'à un nombre fixe k de défaillances. Troisièmement, de multiples échantillons d'états de nœuds sont simulés, et nous calculons les états de chemins booléens de bout en bout. Le nombre de défaillances simulées est ensuite varié pour vérifier le comportement de l'algorithme dans différentes situations. Finalement, l'algorithme d'inférence estime les états des nœuds à partir des mesures simulées de bout en bout, et les estimations sont comparées aux états réels pour évaluer la précision du système de supervision.

Pour réaliser les tests, nous considérons d'abord une chaîne de services réseau composée de 3 VNF. Chaque fonction réseau a entre 4 et 5 instances. Les connexions entre deux ensembles consécutifs d'instances de VNF sont générées de manière aléatoire. L'algorithme de correspondance sélectionne un nombre minimal de connexions permettant de détecter jusqu'à k noeuds défaillants simultanés, et k est fixé à 1 dans ces tests. Ensuite, nous simulons les mesures des nœuds comme expliqué précédemment. Nous faisons varier le nombre de défaillances de nœuds générées de 1 à 3.

La figure 6 illustre la dispersion des probabilités et la matrice de confusion pour les états des nœuds réalisés dans ces tests. Avec un seul nœud défaillant, le système de supervision est capable de les localiser avec précision dans tous les tests. Ces résultats sont attendus puisque la topologie testée remplit la condition suffisante pour la détection d'un seul nœud défaillant. Le nombre de faux négatifs et de faux positifs est donc nul.

Si la condition suffisante pour la détection de k défaillances n'est pas satisfaite, le système de supervision peut toujours les détecter si elles se produisent. Lorsque la condition suffisante n'est pas remplie, il peut exister une ou plusieurs situations (une combinaison de nœuds défaillants) où le système de supervision est incapable d'identifier les défaillances de manière

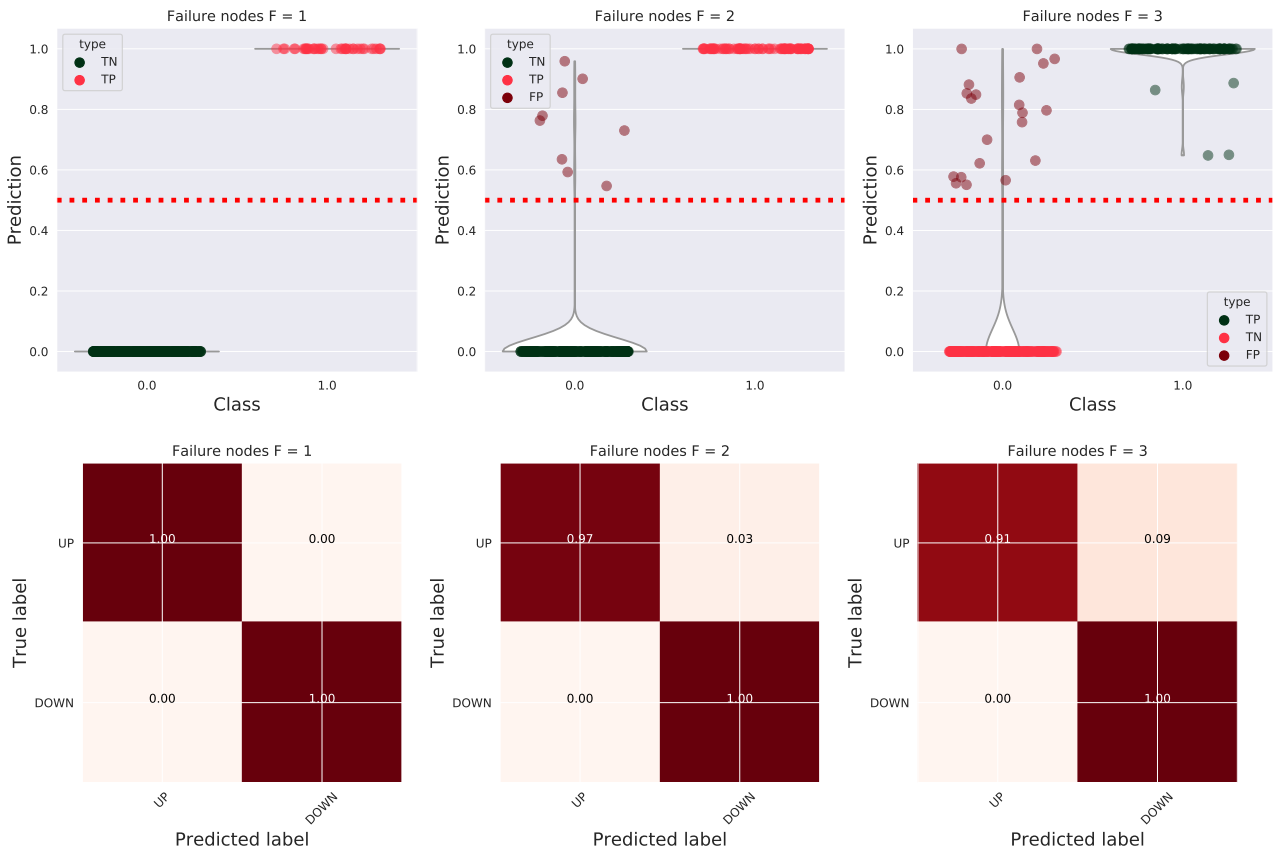


FIGURE 6 – Distributions de probabilité des classes et matrices de confusion (Correspond à la Figure 6.3 dans le manuscrit).

déterministe. Les tests avec deux ou trois nœuds défectueux simultanément confirment cette observation. En fait, avec deux et trois nœuds défaillants, l'algorithme d'inférence les identifie tous. Cependant, le nombre de faux positifs augmente un peu pour atteindre 3% et 9% respectivement.

Conclusions

Pour conclure, les réseaux 5G offriront des capacités importantes en termes de latence, de bande passante et de connexions massives. Cette évolution nécessite l'utilisation de nouveaux paradigmes tels que le SDN et le NFV qui offrent la flexibilité, l'autonomie et la dynamicité dans la gestion et l'orchestration des ressources. Les outils de supervision doivent alors donner une vision précise de l'état des ressources et du trafic. Entre-temps, plusieurs problèmes peuvent être rencontrés en ce qui concerne l'accès aux systèmes de supervision et la disponibilité des ressources. Des outils de supervision s'appuyant sur la tomographie des réseaux pourront apporter une plus value importante à la supervision de ces infrastructures complexes.

Le fil conducteur de cette thèse est la conception de solutions de supervision évolutives pour éviter les problèmes évoqués précédemment tout en garantissant une précision acceptable. Nous avons étudié l'application de la tomographie des réseaux avec cet objectif. L'idée générale de cette approche est d'utiliser des mesures de bout en bout pour déduire les valeurs de métriques sur des liens ou des nœuds auxquels on ne peut pas accéder directement pour des raisons techniques ou administratives. Les solutions qui existaient sont diverses et fiables, mais elles ne sont pas toujours applicables à notre contexte. Nous essayons donc, dans nos contributions, soit de généraliser leur utilisation pour les adapter au problème étudié soit d'améliorer leurs performances du point de vue du temps de calcul et de la précision. Il nous semble qu'elles ont tout leur sens dans le contexte d'un déploiement massif d'architecture virtualisées supportant le déploiement de la 5G dans laquelle la notion de slices déployés à la demande pour exécuter des services est essentielle.

LES CONTRIBUTIONS DE LA THÈSE:

Conférences internationales:

A. Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « Unicast Inference of Additive Metrics in General Network Topologies », *in*: *27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2019, Rennes, France, October 21-25, 2019*, IEEE Computer Society, 2019, pp. 107–115.

B. M. Rahali, J. Sanner, and G. Rubino, « TOM: a self-trained Tomography solution for Overlay networks Monitoring », *in*: *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, 2020, pp. 1–6.

C. M. Rahali, J. Sanner, and G. Rubino, « FEAL: A source routing Framework for Efficient Anomaly Localization », *in*: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

Papiers/journals soumis:

D. Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « The Evolutionary Sampling Algorithm: a genetic algorithm approach for network tomography », *in*: .

E. Mohamed Rahali, Jean-Michel Sanner, Cao-Thanh PHAN, and Gerardo Rubino, « Failure nodes localization in NFV networks using network tomography », *in*: *2017 IEEE Conference on Computer Communications, INFOCOM 2021*, IEEE Computer Society, 2021.

Workshop local:

F. Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « Poster: SDN and Virtualisation: A new boost and opportunity for network tomography », *in*: SDN day 2018, Paris, 2018.

CHAPTER 1

INTRODUCTION

1.1 Motivations

The need for connectivity services is always increasing, with more exigent demands in terms of high bandwidth, low latency, and a high number of connected devices. The evolution towards a new generation of mobile networks able to fulfill these expectations becomes essential. 5G is the new generation of this technology. The promised bandwidth will be much larger than the offered by Long Term Evolution (LTE) technology. Also, the response time has to be very short and can reach one millisecond for some applications. These properties will promote the development of new services that have been classified under three categories [1]: Enhanced Mobile Broadband, Machine Type Communication, and Ultra-low latency, as illustrated in Fig. 1.1.

These services, with their contradictory requirements, must share the same network infrastructure. The new 5G architecture must have the needed features to ensure their cohabitation. Virtualization will then play a central role in this mobile network evolution. Indeed, techniques such as Software Defined Network (SDN) and Network Functions Virtualisation (NFV) allow the flexible creation of virtual networks called *slices* over the same physical infrastructure while respecting the isolation constraints. The slicing concept allows to create multiple levels of resource abstractions and multi-tenancy support. These new features offer huge flexibility in the management of these resources.

One of the associated “costs” of this evolution is the fact that the monitoring task becomes much more challenging. In such an environment, it is not easy to ensure a global monitoring solution because each actor will have his own monitoring system with mutual access restrictions. Thus we need to overcome these limitations while keeping an accurate image of the infrastructure. The use of end-to-end measurements can be a good alternative in such situations. These data can be easily obtained from the traffic collected at the network perimeter without the need to additional access authorizations or specific protocols. Afterward, this information can be analyzed to infer the internal state of the network without making direct measurements. This approach is known by network tomography [2] [3] and it has been well explored in the literature. Most of the treated use cases focus on classic network topologies, but they can be easily adapted for new applications related to virtual network monitoring.

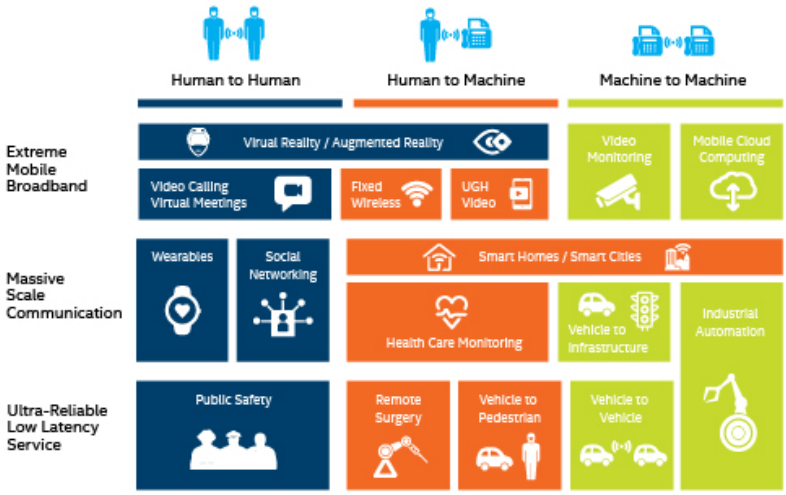


Figure 1.1 – 5G scenarios

1.2 Contributions of the thesis

The main objective of this thesis is to study the applicability of tomography solutions to the supervision of virtual SDN/NFV networks. Nevertheless, most (but not all) of our proposed contributions are not specific to this particular context and remain applicable to other monitoring use cases.

In our first contribution, we studied the use of unicast measurements for the inference of additive link metrics. Contrary to most of the proposed solutions which focus on tree topologies, our proposals can be applied to general ones. We have studied in particular a method based on the EM algorithm that we called k-paths. This approach is pretty accurate but it is also characterized by a high computational and memory complexity which limits its use on large topologies. To overcome this problem, we proposed the Evolutionary Sampling Algorithm (ESA) algorithm, which is inspired by the steps of the EM approach plus the principals of genetic algorithms. The results show a significant improvement compared to the k-paths method. Then, we proposed the Fast-ESA (F-ESA), which is an adapted version for much larger topology dimensions. It looses a little bit of precision compared to the ESA algorithm, but it is much faster. We evaluate the ESA approach with additive metrics inference tests. Meanwhile, the principle remains applicable for other ones such as the available bandwidth or Boolean metrics, as we illustrate in this document.

In the second contribution, we use a trained Neural Network architecture as an inference tool to deal with the same problem. The model learns from only with a few simulated data.

The training time is very short which gives the possibility to repeat it according to the network updates. This method gives a good accuracy level, similar to that of our statistical technique ESA.

The third contribution focuses on the monitors' placement problem, the task that logically precedes the inference one. We have studied a particular case where only cycle paths are used. The motivation behind this choice is to avoid the constraints related to the synchronization between nodes. Moreover, the network programmability offered by SDN allows the rather easy deployment of such customized type of probing paths. This task is formulated as a set covering problem and we proposed two approaches to solve it, an optimal algorithm and a heuristic one. We test our solution through an anomaly detection scenario with a proof of concept implemented over the Mininet emulator and a P4 implementation of source routing.

In the fourth and last contribution, we studied the anomaly detection problem in NFV networks. We apply the principles of Boolean network tomography, where we consider that each node can have only two states: up or down. We give the necessary and sufficient conditions to detect k simultaneously failed points in a virtual network topology. Then, we apply the ESA approach for the inference of those Boolean metrics.

1.3 Organization of the manuscript

The first chapter of the thesis is an abstract in French. Chapter 2 overviews the main concepts of 5G networks, the motivations, and the key enablers and challenges. We explain our particular interest on the emerging monitoring issues. We propose a classification for the approaches of network supervision according to the used technologies, the targeted network environments, and their applications. Then we survey the related works in network tomography, since the main contributions of this thesis belong to this category.

In chapter 3 we introduce our solution for the inference of additive metrics called ESA. It has been compared with an existing similar solution using simulations. Then we present F-ESA, a faster variant, actually an adaptation of ESA to the case of large topology dimensions. We also briefly discuss the applicability of our approach to non-additive metrics.

Chapter 4 deals with the same inference problem, but following a different approach based on Machine Learning using Neural Networks.

In chapter 5, we study the complementary monitors' placement problem using probing cycles. We formulate it as a set covering task and solve it following optimal and greedy approaches. Then, we give a proof of concept implementation to test the two procedures.

Chapter 6 provides a formulation of the problem of detecting failed nodes in virtual NFV networks. We studied the network topology properties related to the goal of localizing a maximum number of simultaneously failed nodes.

Finally, Chapter 7 concludes this thesis and gives the outlines of some associated research perspectives.

CHAPTER 2

CONTEXT

2.1 5G: motivations, key enablers and challenges

2.1.1 5G motivation

The arrival of 5G networks would be a good opportunity to accelerate and enhance connectivity services. With its promises of high throughput, massive connectivity and low latency, more innovative services become feasible. Behind these expectations, multiple technical challenges should be solved to fulfill these high demands.

The 4G network architecture was designed to give generic support to all the services despite their different requirements. To satisfy these increasing traffic demands, network operators should optimize the usage of their resources. Indeed, the 5G network architecture should evolve to afford the needed flexibility and scalability. One of the key features in 5G is the capacity to design specific network infrastructure to satisfy the different needs of the business offers, relying on new paradigms like NFV and SDN. These concepts aim to offer more flexibility and scalability while decreasing the Capital Expense (CAPEX) and Operational Expense (OPEX) costs of the network infrastructure. The classic network functions can be implemented as a software that can be executed over generic hardware. Thus, the network resources can be dynamically allocated and automatically scaled according to the clients' demands. This allows the operators to host multiple services with varied requirements on the same infrastructure.

The 3rd Generation Partnership Project (3GPP) [21] standard has specified multiple scenarios, the mechanisms, and the architectures for network sharing. The concept of network sharing is continuously evolving to the support of multi-tenancy, until reaching the modern network slicing idea. As introduced by Next Generation Mobile Networks (NGMN) [22], network slicing aims to offer customized services and multi-tenancy support on a shared network infrastructure while ensuring isolation and lifecycle management [23]. Thus multiple network slices can share the same network function or sub-network. Besides, these resources can be located in different geographic placements and managed by multiple infrastructure providers.

All these features enable operators to efficiently manage their offers and increase their benefits. Meanwhile, the price of this flexibility can appear in the supervision of these co-located services with the intervention of multiple actors. The sub-networks composing the network slice

can be managed by multiple providers, and each one may have its own private monitoring system. Indeed, it is difficult to find the root cause of a Service Level Agreement (SLA) violation in a network slice.

The new network architecture does not only promises high capacity and availability, but also automatic and dynamic solutions for network orchestration. The main objective is to reduce the human intervention and automate the life cycle management of the network services which leads to the idea of Self-Organizing Networks (SON) [24]. To achieve this objective, the orchestration system should have an accurate image of the network resources' state and the efficient tools for fault diagnosis and remediation actions. The monitoring system should be characterized also by a high scalability and flexibility as the resource management system. To summarize, future monitoring systems should satisfy these main criteria:

— High accuracy: To ensure an efficient management of the network, the orchestration system should have an accurate image of the available resources and a good prediction of future demands.

— Flexibility: The new monitoring tools should support the particularities of the new network architectures adopted in 5G. The diversity of the supported services will introduce multiple and heterogeneous Key Performance Indicators (KPIs). The presence of multiple actors in a shared network infrastructure will introduce more complexity for the monitoring system.

— Programmability: Thanks to the SDN technology, future networks will be marked by high dynamicity and programmability. Indeed, the monitoring tools should have the ability to deploy and adapt the monitoring strategy according to the network changes.

2.1.2 5G drivers

The new 5G architecture adopts some paradigms to fulfill its high expectations. The main key enablers of 5G networks are the SDN and the NFV concepts. Thanks to these technologies, network operators can manage efficiently their network resources. In this section, we detail these new concepts and why they become the most important ingredients for the success of this technology.

2.1.2.1 Software Defined Network (SDN)

The SDN is an emergent network architecture that offers more programmability and dynamicity to the network control. The main advantage of the SDN architecture is decoupling the control plane from the data plane.

This feature adds more programmability to the network devices. Thus, the traffic forwarding logic can be easily adapted by a centralized unit called the SDN controller. Fig. 2.1 illustrates

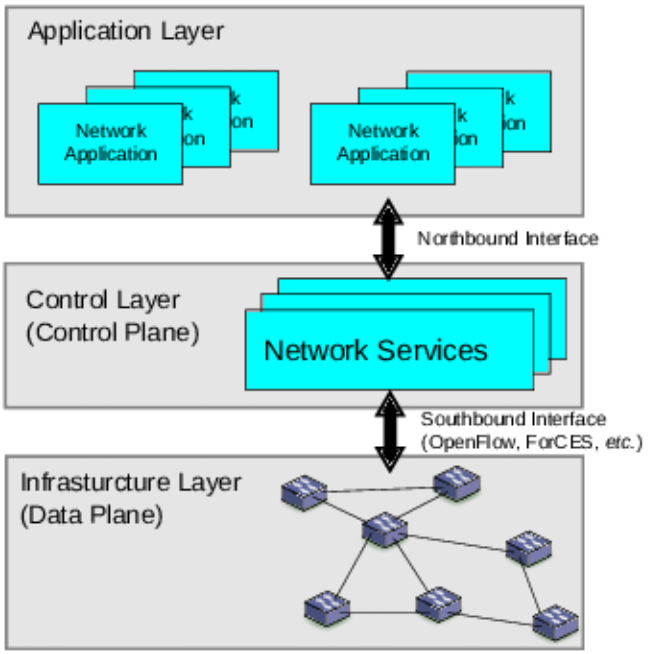


Figure 2.1 – SDN architecture

the main components of the SDN architecture which is composed essentially by three layers: the application layer, the control layer, and the data plane:

- Application layer: It includes the software applications running over the network. These applications communicate with the Northbound RESTful APIs of the controller to demand the needed network resources.
- Control layer: It is a software centralized entity that has a global view of the network resources. It communicates with the application layer via the Northbound APIs to receive the demands for network resources. Then, the controller translates these high-level requests into the equivalent network configurations in the concerned devices via the Southbound interfaces.
- Data-plane layer: It is represented by the network resources, including the routers, the switches, and all the network instruments. These devices are programmable via the SDN controller and can be physical or virtual.

The network controller has a central role in the SDN architecture. The applications send their traffic management logic to the controller which translates it into network configurations. The controller communicates with the network devices via the Southbound interfaces. OpenFlow (OF) [25] is one of the most used Southbound interface protocols. It was proposed by the Open Networking Foundation (ONF) and becomes the most associated protocol to SDN networks. It is worth noting that other protocols can be used for Southbound interface communications like NETCONF, BGP, and SNMP to support legacy networks in the SDN architecture. A great effort

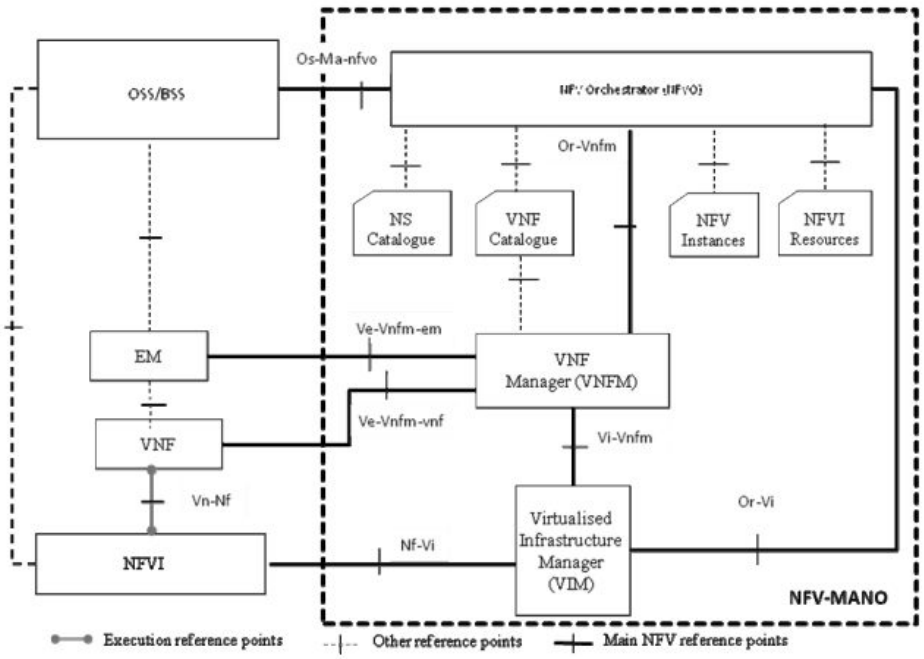


Figure 2.2 – MANO framework

is being made by the standardization organisms and all the parties involved in the telecommunication field to propose standard solutions to evolve toward a universal programmable network infrastructure [26]. For example, the OpenConfig working group involves multiple operators to propose standard interfaces and vendor-neutral APIs to manage and configure the networks based on YANG data models.

2.1.2.2 Network function virtualization

The NFV concept becomes an important key enabler for 5G since it allows to implement network functions as a software and to run them over commercial generic hardware. As a result, network resources can be managed efficiently, taking advantage of the virtualization techniques adopted in IT. According to the traffic users' demands, the allocated resources can be scaled up or down, relocated, or removed. Either the Core Network (CN) functions or the Radio Access Network (RAN) functions can be virtualized [27].

The European Telecommunications Standards Institute (ETSI) defines the Management and Orchestration (MANO) framework for network function virtualization. Fig. 2.2 illustrates the main components of the MANO framework architecture:

— The Virtual Network Function (VNF)s: They represent the network functions software implementations. They are deployed in the virtual infrastructure.

— NFV Infrastructure (NFVI): It includes the physical layer, the virtual resources, and the

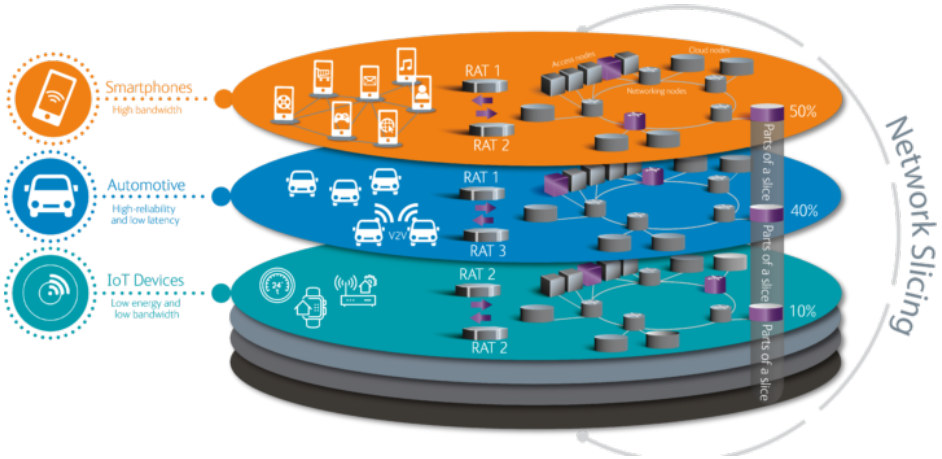


Figure 2.3 – Network slicing concept

software needed to host the VNFs.

— Management and Orchestration (M&O) system: It is responsible for the life-cycle management and the orchestration of the network services, the NFVI, and the VNFs. It is composed of the NFV Orchestrator (NFVO), the VNF Manager (VNFM) and the virtualized infrastructure manager (VIM).

SDN and NFV are complementary and they provide together a great potential for the development and innovation of new services. With the NFV concept, most of the network functions can be virtualized and seen as scalable blocks that can be dynamically replaced according to the traffic changes. Thus, the SDN can serve the NFV as a tool that provides the interfaces to program the network.

2.1.2.3 Network slicing

Thanks to the flexibility and programmability offered by SDN and NFV, operators can manage efficiently their resources. Indeed the physical infrastructure can be shared between multiple vertical business entities or application providers.

The network slicing paradigm will make the network architecture more flexible to support varied business offers with their specific performance requirements. Relying on the strength of SDN and NFV, it becomes the backbone of the progress of the 5G technology. The idea of building a logical network over a shared physical infrastructure, known as an overlay network, is quite old. Overlay networks enable to combine network resources from multiple domains. However, the requirements for 5G networks in terms of programmability and autonomy are much higher. The network slicing architecture should satisfy additional criteria like automation, isolation, and programmability.

Fig. 2.4 illustrates the three layers involved in network slicing: the service instance layer, the

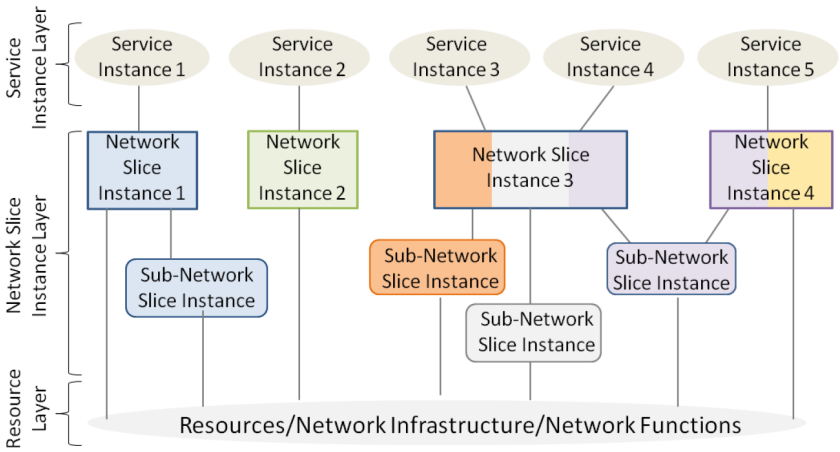


Figure 2.4 – Network slicing layers

network slice instance layer, and the resource layer. The network slice instance represents the network resources required for a customized class of network services. It can be composed of multiple sub-networks. A sub-network instance can be composed of one or multiple network functions and the needed network resources for them; these resources can be either physical or logical. It is generally defined by a sub-network descriptor/blueprint and may be shared by different network slices. A sub-network can be dedicated to a single slice or shared between multiple ones. Indeed, the network resource sharing concept is present at different levels of the network virtualization process which enables still more flexibility.

An important feature of the network slicing architecture is its ability to deal with multiple abstraction levels. Thus, the resources composing a network slice can be provided by multiple infrastructure providers, collocated or not. Also, the slice can be, totally or partially, allocated to another client. All these features allow operators to efficiently manage their offers and increase their benefits. Meanwhile, the infrastructure management and the supervision of these co-located services with the intervention of multiple actors can be much more complicated. The sub-networks composing the network slice can be managed by multiple providers, and each one should have its private monitoring system. Indeed, it is difficult to find the root cause of an SLA violation in a network slice.

A network slice must transparently provide end-to-end connectivity while involving heterogeneous network resources belonging to the RAN and CN. The slicing concept is more convenient to apply for the CN part. The VNFs composing this part require essentially computing capacity and memory which are easy to scale when needed in a cloud environment. Also, the connection between the data centers hosting the VNFs is essentially ensured by optical links that have a high bandwith capacity. For the RAN part, the concept of slicing remains applicable but the constraints differ from the CN due to the scarcity of radio resources [28].

2.1.3 5G challenges

Network slicing is recognized as a crucial element in the development of 5G networks by allowing the migration from a one-fits-all architecture to more customized ones. To reach the expected level of flexibility and autonomy in future networks, several challenges arise regarding the control, the supervision, and the management of network resources.

2.1.3.1 Network resources allocation

A network slice can be seen as a virtual network deployed over a physical network with specific constraints for the links and nodes capacities. Thus, the resource management in network slicing is quite similar to the classic virtual network problem where we try to optimize the placement of VNFs while optimizing the available resources in terms of bandwidth and processing capacity [29]. Meanwhile, there are additional constraints specific to network slicing like isolation and high end-to-end QoS requirements. Besides, future wireless networks will be characterized by high dynamicity and variability. Thus, the resource allocation strategy should be sufficiently flexible to handle fast and frequent adaptations with a minimum cost.

2.1.3.2 Network monitoring in the context of 5G

To follow the impressive progress promised by 5G networks, the monitoring system should be characterized by high scalability and flexibility as the resource management system. In such heterogeneous networks, the deployment of a centralized monitoring system can be constraining. However, it is possible to design a lightweight scalable solution based on end-to-end measurements. For example, if there is a failure point in the shared sub-network or virtual function, all the services that use this node can have some miss-behaviors. It is possible to infer the state of the intermediate nodes from the end-to-end metrics by taking advantage of the correlations created by the shared nodes. This is introduced in the next subsection.

2.2 Network monitoring: state of the art and new challenges

Network monitoring has always been considered an active field of research. The varied services deployed in today's networks need high Quality of Service (QoS) requirements which differs from a service to another [30]. Network monitoring can have different purposes. The verification of equipment configuration is an important usage of the monitoring tools. Especially in an SDN/NFV environment, the number of configurations ordered from the controller to the network devices will increase since one of the most principal features of SDN is the simplification of equipment configuration and application deployment. In such an environment, network

application deployment will be more automatic and the verification of these operations needs to be flexible and reliable. To guarantee the reliability of the monitoring operation, the choice of the appropriate approach is crucial. There is a large spectrum of network monitoring solutions that differ from each other in the used technologies, the suitable usage, and the consumed resources. So the comparison between the monitoring tools and approaches needs to specify the context and the use case.

2.2.1 Classic IP-networks monitoring

Conventional IP-networks are usually equipped with multiple monitoring and diagnostic tools. Network equipment providers offer operators the needed solutions to check the status of the equipment and collect statistics on the traffic passing through. This information can be obtained by polling protocols like Simple Network Management Protocol (SNMP) [31] which is widely supported in IP networks.

The SNMP monitoring system is based on two principal elements: the SNMP Manager and the SNMP agent. The Manager is the module that communicates with the agent to request the needed information. The agent is represented by a software integrated within the network equipment. Its main role is to collect information about the device and the network traffic, report events to the Manager and respond to the queries. The monitoring information is stored in a local database called the Management Information Base (MIB). This data structure is represented in tree form, where the leaves represent the objects that contain the stored data. Each object can be identified through its Object Identifier (OID).

Traffic sampling is another mode of network monitoring. Netflow [32] and Sflow [33] are two technologies for traffic sampling in IP-networks. These technologies are supported by most of the commercial network routers. Netflow makes random sampling of the traffic that passes through the equipment. Then, the packets are analyzed to increment the flow counters. The collected information about the traffic flows can be used to compute the flow matrix, for example. This technique can collect information only about large flows called “elephants” and neglect the smaller ones. Sflow makes equitable sampling for all the flows passing the equipment. Thus, it can detect also the small flows but this needs more resources in the device.

2.2.2 SDN-based monitoring solutions

The SDN concept offers flexibility and automation for network configuration and application deployment. To ensure a good performance of these services, the monitoring techniques should have the same level of flexibility and agility to have reliable information about the network state. In SDN networks, the number of configuration orders will increase. These orders can be sent automatically via the controller. The different operations should be verified by the

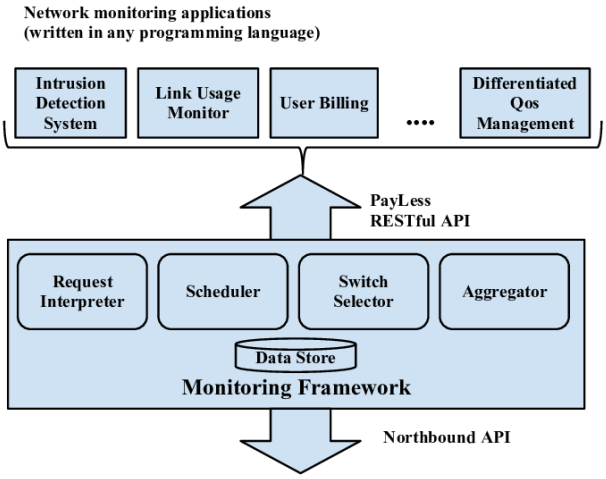


Figure 2.5 – Payless architecture

monitoring system. Classic monitoring methods could have some problems to deal with this increasing agility. The monitoring system can also benefit from the flexibility offered by the SDN to design adaptive and programmable solutions. The different supervision operations, from data collection to their analysis, can be improved thanks to the SDN features.

In [34], the authors overview the impact of the SDN concept on the different monitoring steps. Most of the proposed works rely on the OpenFlow protocol to design the monitoring solutions. For example, paper [35] proposes the HONE monitoring framework to minimize the traffic measurement data. It is based essentially on a software agent running over a host and a module that interacts with the network devices and makes the needed requests. This framework proposes two techniques. The first one called “lazy materialization of fine-grained statistics”, where the collected network flow statistics are pre-processed locally on the hosts and stored in a customized data structure. Then, the requests between the SDN controller and the agents integrated within the hosts can be optimized to reduce the data transfer overhead. The second proposed technique allows the partitioning of the data processing between the hosts and the controller, which reduces the data transfer and the controller overhead.

Paper [36] introduces the OpenSketch SDN monitoring framework. The solution includes additional components to integrate within the SDN controller and the switches to enhance the performances of the monitoring system. Depending on the traffic stability and the characteristics of the present flows, the controller orchestrates the different measurement tasks in the switches depending on the resources availability (CPU, bandwidth, and memory). This solution proposes different features for SDN network monitoring, but it requires important modifications on the switches and the controllers.

OpenNetMon [37] and Payless [38] are two SDN monitoring solutions based on Openflow features. OpenNetMon [37] relies on the control messages between the controller and

the switches to identify the existing flows in the network. PacketIn message (it is an Openflow feature) notifies the controller when a new flow arrives. A FlowMod message is sent by the controller to the switch to install a new path for a traffic flow. FlowRemoved message is sent by the switch when a flow is removed. It contains supplementary information about the removed flow like packets and byte counts. The controller sends periodic queries to collect statistics about each flow with the FlowStatic request. The polling strategy of each flow is adapted according to the stability of the network traffic.

The Payless [38] framework proposes Northbound restful APIs to communicate with the different applications and adapt the polling strategy to their needs as described in Fig. 2.5. These adaptations concern the polling frequency and the selected requested switches to find a trade-off between the measurement overhead and a good accuracy.

In [39], an OpenFlow monitoring solution is proposed to enhance fast recovery for link/node failures. A monitoring function is integrated within the OpenFlow switches to pre-process the monitoring messages and reduce the load on the controller. The TinyFlow framework is proposed in [40] for traffic management in data center networks. Traffic statistics are contentiously recorded to detect the “elephant” flows and then to divide them into smaller flows called “mices”.

The Programming Protocol-independent Packet Processors (P4) programming language can be an alternative to OpenFlow in SDN networks, as showed in [41]. The authors propose the FlowSpy framework based on the P4 language to design customized switches that can handle more monitoring tasks compared to OpenFlow switches like Open vSwitch (OVS). Then, the monitoring tasks are partitioned between the switches. The assignment process is formulated as an Integer Linear Programming (ILP) problem.

2.2.3 Monitoring in virtual networks

2.2.3.1 Cloud monitoring

Network function virtualization is based essentially on the cloud computing concept and technologies. As a consequence, virtual networks monitoring can benefit from the adopted techniques in cloud computing.

Papers [42], [43] and [44] overview the different aspects of cloud networks monitoring. The cloud model defines seven layers as proposed by the Cloud Security alliance: facility, network, hardware, OS, middleware, application, and the user layer [45]. The notion of layers is very important to take into consideration when setting up a monitoring system for a cloud architecture. The choice of the layer to collect the measurements is important for the efficiency of the operations. Indeed, some events or accidents in the network can only be detected or diagnosed with a careful choice of the supervised layer.

The metrics that can be supervised in such environments can be classified into two cate-

gories as proposed in [42]: computational and network metrics. Computational metrics concern the supervision and the management of physical resources, including CPU, memory or bandwidth. The network metrics focus on the connection quality between data centers, such as delay, loss rate, and jitter. The monitoring system in a cloud environment must satisfy some properties specific to this context. An important one is the scalability with dynamic and fast allocation of the needed resources. The monitoring system must manage the variations in the volume of the uploaded data and its processing. Another important feature is the elasticity. Since the cloud infrastructure hosts diverse applications with varied needs, the monitoring system has to handle the dynamic updates and the varied needs. Last but not least, the monitoring solution should be sufficiently autonomous. As the cloud must be characterized by a high degree of autonomy to respond to the client demands, it must also be able to ensure self-management and to effectively detect misbehaviour to take the necessary corrective measures.

2.2.3.2 NFV monitoring

The success of the cloud model has motivated players in the telecommunication market to adopt this approach to evolve network infrastructures. Therefore, the objective is to design a model that offers more scalability, autonomy, and elasticity to network services. The main idea is that the network functions are implemented as a software that can be hosted in generic cloud infrastructures as defined by the Network Function Virtualisation (NFV) concept [46]. Thus, the NFV model is based on the principles of virtualization by adopting the same properties and technologies used in the cloud as described in Fig. 2.6, particularly for the monitoring system [47]. Meanwhile, the NFV paradigm presents additional challenges since a network service is always composed of a VNF chain that must be managed entirely. The constraints and requirements in this case are more exigent as the failure of a single function will imply the failure of the whole service.

Indeed, an effective monitoring system for NFV networks should have the necessary tools to collect measurements in the different network layers and from the network infrastructures hosting the network functions to the service layer. In the MANO framework, the Virtualized Infrastructure Manager (VIM) is responsible for the management and supervision of the Network Functions Virtualisation Infrastructure (NFVI) hosting the network services. Indeed, the VIM collects information about the network state and sends them to the orchestration unit.

The orchestrator can receive a huge volume of data uploaded from several entities of the NFVI. Indeed, a pre-processing of the data is essential in this case to aggregate the collected data, analyze it, and transmit to the orchestrator only the useful information. It is also possible to consider distributing the orchestration to lighten the load on the main orchestrator. The VIM can be implemented using data center management technologies such as OpenStack [48]. Several monitoring tools can be integrated or interfaced with OpenStack. OpenStack Ceilometer is a

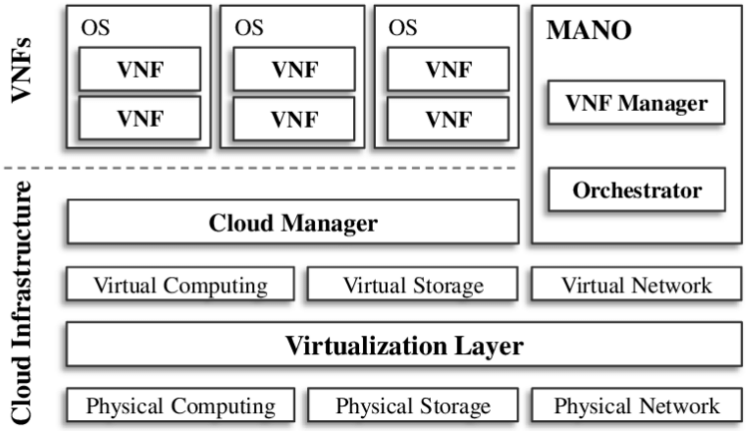


Figure 2.6 – Architecture for NFV

telemetry module that collects metrics on the status and the usage of both physical and virtual resources. This tool can be easily interfaced with other modules or plugins for monitoring such as Zabbix, Nagios, Zennoss, Monasca, and Gnocchi. These tools can be used to monitor the VMs hosting the VNFs. The network part that provides the connection between the different NFVI components must also be supervised. The NFVI components can belong to multiple domains and can be controlled via an SDN solution that usually provides the needed monitoring tools. For example, the OpenFlow [25] protocol offers multiple features to collect flow statistics. The OpenDaylight SDN controller offers also some APIs for metrics collection [49].

In [50], the authors propose the T-NOVA solution for NFV monitoring. The main component of the framework, called the monitoring manager, is integrated at the VIM level. It is charged to aggregate the metrics remounted from the NFVI via OpenStack and OpenDaylight telemetry and statistics APIs. The framework offers also an agent that can be integrated within the VNF image to collect more metrics proposed by the VNF descriptor. This agent can be either pre-integrated in the VNF image, installed while the VNF deployment or removed according to the user's preference.

Multiple proposed solutions like NFV-Vital [51] or CloudScale [52] for NFV monitoring consider the supervision of VNFs as a classic cloud application. It is often proposed to use measurements of hardware resources utilization (CPU, memory, and bandwidth) to identify anomalies. Meanwhile, several problems cannot be detected following this approach and require a more in-depth analysis of the network traffic for the concerned VNF. For example, [53] describes the NFVPerf solution for VNF performance evaluation and bottleneck detection. The proposed framework can be used for VNF monitoring to make port mirroring for a selection of VNFs communications and computes application-layer metrics like throughput and delay. These metrics are aggregated in a centralized module to be analyzed via a bottleneck detection algorithm.

In [54], the authors propose a monitoring architecture for NFV networks. The proposed

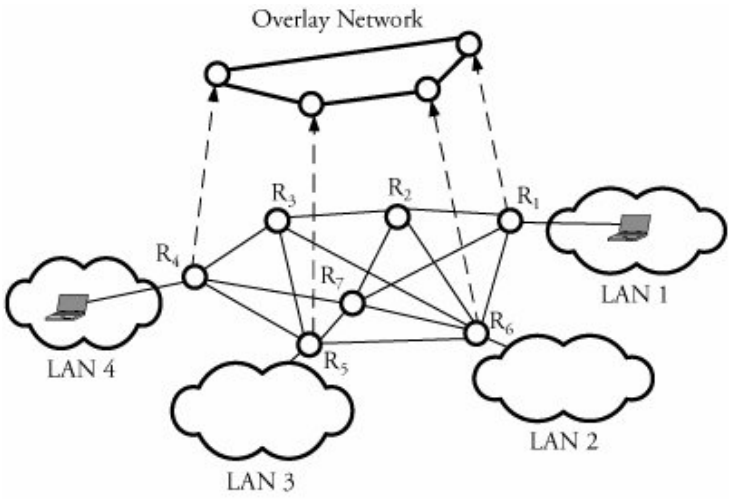


Figure 2.7 – Overlay network example

framework allows a dynamic selection of nodes in the NFVI to sample the traffic and remote flow statistics to an anomaly detection module to make the analysis. Paper [55] proposes the z-TORCH solution for the optimization of the MANO system while reducing the monitoring cost. The monitoring of each VNF is essentially based on the KPIs provided by its descriptor. Then, an unsupervised machine learning technique is used for the profiling of each VNF. Afterwards, the VNF placement algorithm search for an appropriate location for the VNFs, taking into account their profiles and the temporal variations.

In [56], the authors propose a machine learning approach for VNF anomaly detection. The proposed solution has three main functions: anticipating failures by detecting the first signs of SLA violations, detecting SLA violations, and finding their root causes. This tool helps the network administrators to make efficient and fast troubleshooting and to take remediation actions like rebooting a VM or scaling the allocated resources. The approach is based on machine learning models and supervised learning. The training dataset is formed by collecting data remounted from the VMs. Fault injection is used to emulate anomalies since the collected data represents mostly the normal state.

2.2.3.3 Overlay networks monitoring

Monitoring the functioning of individual nodes in a virtual network is not enough to ensure the overall operability of the services. Reporting metrics on the status and the usage of the allocated resources to each VNF does not enable the detection of some problems related to end-to-end performances and the cohabitation of different services on the same virtual infrastructure.

Therefore, it is always recommended to perform end-to-end QoS performance metrics. Af-

terwards, when a performance degradation is detected, it is possible to do a more in-depth diagnosis and to collect measurements. This enables to supervise the functioning of suspicious nodes and links in the network infrastructure.

The idea of supervising and managing an abstract view of the network is relatively an old concept known by overlay network monitoring (see the illustration in Fig. 2.7). Multiple connectivity applications can benefit from overlay network monitoring such as peer-to-peer routing. The measurement techniques and the studied metrics differ from monitoring separated nodes [57]. For example, paper [58] describes the Iso-bar monitoring system to estimate the distance between peers in an overlay network without any knowledge of the underlay layer. The solution gives a process to select the appropriate placement for deploying the needed monitors allowing an accurate estimation of the distance between the hosts in different administrative domains.

Supervising all the links in a network can sometimes be cumbersome and costly. Indeed, the authors in [59] propose to make limited link measurements to reduce the overall cost of the monitoring operation. This process is formulated as an optimization problem, where the objective is to select the optimal set of accessible links from the underlay layer and a limited set of path measurements enabling a good estimation of overlay link metrics. Then, the necessary instructions to infer the unobserved overlay links are provided.

Paper [60] deals with the scalability issue in overlay network monitoring. The problem is formulated as a minimum set covering and solved following a greedy optimization approach. The solution selects a minimal set of overlay paths enabling to infer the rest of them while ensuring a good accuracy.

Paper [61] proposes an event monitoring infrastructure for overlay networks called MOON. This solution describes a set of techniques and algorithms allowing efficient event detection. The authors propose a geographic clustering of the network nodes. Thus, the proposed framework enables a distributed event tracking which reduces the monitoring latency as well as the scalability problems.

2.3 Network tomography

The management of a large multi-domain network can be a difficult task due to the technical complexity. Usually, monitoring systems are limited by the authorizations access to the concerned administrative domains, the lack of protocols, and the network resources overhead. These issues make it complex to design a monitoring solution with a holistic view of the network. Thus, more customized supervision tools are needed for these situations, in order to estimate the internal state of a domain from external measurements.

The inference of internal network metrics from partial end-to-end measurements is known as network tomography [2] [4]. The origin of this terminology lies at the image processing field,

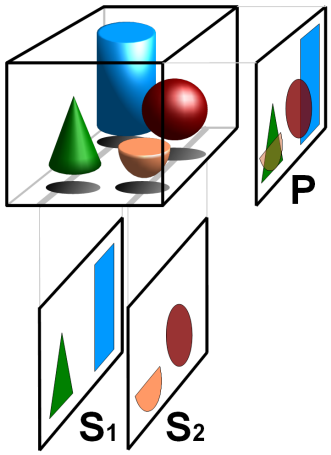


Figure 2.8 – Image reconstruction / Image tomography

where it represents a technique to reconstruct the volume of an object from a series of measurements taken outside the object, as illustrated in Fig. 2.8.

Similarly in network monitoring, network tomography aims to construct a complete image of the internal network state from a partial view coming from external measurements as illustrated in Fig. 2.9. It has been an active research field over the past two decades. These techniques do not require additional access authorizations for the equipment or specific protocols to collect statistics. The end-to-end measurements can be either collected passively by sampling the existing traffic flows, or actively by injecting specific monitoring traffic that can be routed inside the network as regular users traffic.

The first proposed solutions focus on the inference of internal parameters of the network (link or node metrics) from external or end-to-end measurements. The supervised paths and the monitoring strategy are considered as inputs of the model to derive the best estimation of the unknown variables from the available information. The link/node metrics are often considered as random variables following a probabilistic distribution. Then, multiple statistical methods can be applied to estimate the distribution parameters. The proposed estimators vary according to the properties of the applied probing method and protocols (Unicast/Multicast, TCP/UDP, etc).

Recently, more attention has been accorded to the design of monitoring schemes, probing paths, and network topological conditions enabling an accurate estimation of the internal link metrics. The adopted approaches can be either statistical [62] or algebraic [63] as it will be detailed in the next paragraphs.

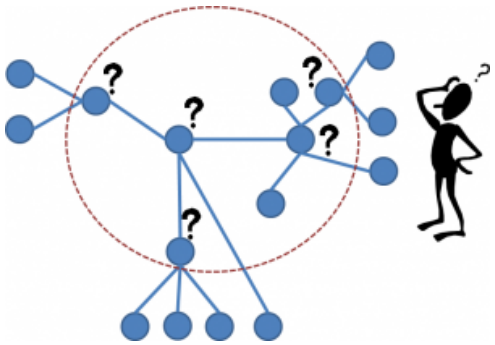


Figure 2.9 – Network tomography

2.3.1 Metrics

An important step in any monitoring operation is the choice of the data to be collected and its acquisition using the dedicated tools. Network tomography solutions can be classified according to the used metrics and the targeted objective. Therefore, we can distinguish two main categories of solutions. In the first one, the objective is to infer inaccessible metrics on the network elements (links or nodes) composing the network from end-to-end measurements [62]. In the second category, the objective is to estimate traffic intensity on the paths between the transmitters and the receivers based on limited measurements of the traffic on the intermediate links [7].

For estimating the parameters on individual links/nodes, the path measurements are essentially the loss rate, the delay, or the jitter. The loss rate between two nodes can be obtained by counting the packets sent and received between the source/receiver nodes. A packet can be dropped for example when the waiting time in the buffers exceeds the allowed threshold or in the case of node failures due to traffic overhead. For delays, the sending and the receiving time must be recorded for each packet while ensuring a fine clock synchronization in the concerned nodes. A delay on a path includes the propagation delay on the links, the queuing and the processing time in the routers.

For origin-destination traffic intensity inference, the input measurements are the flow packets counts collected from the routers. Technologies like Netflow or Sflow can be used to get such data through the agent deployed in the equipment. The Openflow protocol has also its mechanisms to collect traffic flow statistics and remote them to the SDN controller. Then, given the list of the senders and the receivers, the goal is to estimate the consumed network bandwidth between them. These estimations present the origin-destination traffic matrix.

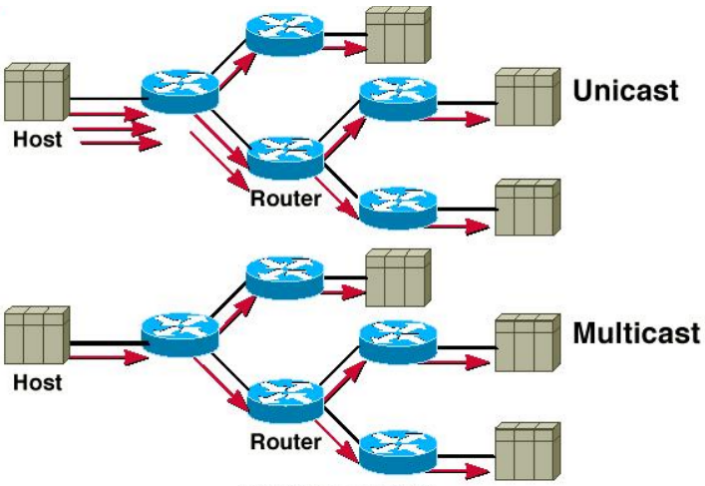


Figure 2.10 – Unicast vs Multicast

2.3.2 Statistical network tomography

The randomness of the studied metrics encourages the adoption of statistical approaches. Generally, most of network tomography problems can be modeled as follows: we assume that the relation between the internal network parameters at time t, denoted by X_t and the external measurement Y_t is defined by a transformation T, where $T(X_t) = Y_t$. Thus, the main objective is to solve the inverse problem by finding or approximating T^{-1}. The proposed tomography solutions address this problem from multiple points of view and following different mathematical approaches. Usually, X_t is considered as a random vector with some probabilistic distribution f with vector parameter θ. Then, the goal is to estimate the unknown parameter θ from the observations Y_t. For example, for additive metrics like the delay or the logarithm of packet acceptation rate (1 - loss rate), the problem can be modeled by a linear transformation as follows:

$$AX_t = Y_t + \varepsilon. \tag{2.1}$$

Where A is a Boolean matrix representing the probing paths and ε is a noise vector. More details about the construction of this linear equation system are given in next chapter. Most of the proposed works consider a simpler model where ε is ignored because of the difficulties in capturing its characteristics. Finding an exact solution to this problem requires that A is full rank, which is not usually the case. Multiple statistical techniques can be applied to approximate the unknown parameters such as maximum likelihood estimators with the expectation-maximization (EM) algorithm, least-squares methods, bayesian estimators, etc.

For the measurement collection, two main modes can be distinguished: unicast and multicast. For unicast communications, the sender transmits each packet to a single receiver. In the multicast mode, the transmitter can send the same packet to a group of nodes simultaneously.

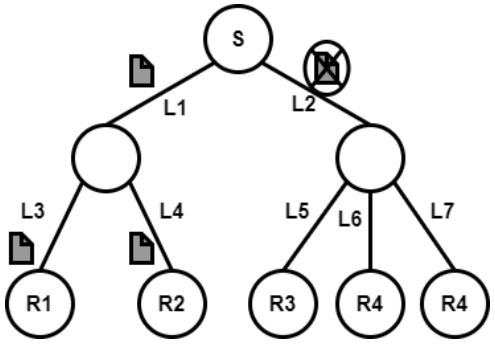


Figure 2.11 – Multicast traffic example

The original packet is duplicated at the intermediate nodes until reaching the final destinations forming a multicast routing tree as shown in Fig. 2.10.

2.3.2.1 Multicast inference

The monitoring data collected with the multicast-based mechanisms has some interesting statistical properties. The packets duplication introduces correlations in each branch of the tree. For example, if a packet is lost in a node or an intermediate link, all the nodes issued from this branch will not receive it. Based on these properties, multiple statistical estimators have been proposed to infer the characteristics of internal link in multicast trees.

The inference of individual link loss rates is one of the most efficient applications of Multicast probing. When a packet is sent from the source at the root of the tree, we can often know the exact link or node where the packet was dropped. Lets consider the simple example described in Fig. 2.11. If a packet is sent from node s and lost at link l_2, it will be received only in R_1 and R_2. Then, based only on the receivers monitoring data, there are only two possible scenarios: either the packet is lost at l_2 or simultaneously at l_5, l_6, and l_7. The second scenario remains very unprobable, especially in the context of highly-reliable networks. For example, if we consider that the link probability loss is equal to 10^{-4}, the joint probability of the simultaneous packet loss in three links will be 10^{-12}. So the first scenario is by far the most likely one. Based on these kinds of observations and properties, multiple sophisticated statistical estimators have been proposed to estimate the loss on intermediate links in a multicast tree from detailed packet traces.

The MINC project is an early initiative for applying Multicast in network tomography. Paper [62] describes a maximum likelihood estimator (MLE) for link loss rates inference. Some assumptions are made about the loss rate model. Indeed, the authors consider independent Bernoulli losses at the different links. The proposed estimations are very accurate and require only a limited number of probing packets.

In [5], the authors studied the same problem described in [62]. However, they assume the

presence of missing data in some receivers which prevents the application of the same estimator. Thus, they use the conventional Expectation-Maximization (EM) algorithm, usually applied with missing data problems, to approximate the solution that maximizes the likelihood of the observed data. Paper [64] studied the application of multicast probing in general network topologies. The main idea is to cover the network topology with multiple multicast trees. Then, two approaches have been proposed for the inference of loss rates. In the first algorithm, each tree is treated separately and the loss rates are computed with the estimator proposed in [62]. Then, the loss rate for each link will be the average of the outcomes from the different trees. The second algorithm applies the EM method. It gathers the data from all the multicast trees as input and computes the loss parameters that maximize the likelihood of the observations. Similar estimators have been proposed for link delay based on the multicast properties under some assumptions like spatial and temporal dependencies between the link delay distributions [65].

2.3.2.2 Unicast inference

Tomography techniques based on multicast have shown a strong efficiency thanks to their statistical properties, but their application remains limited as multicast support is not always guaranteed in existing networks. In this situation, the adoption of unicast probing remains more practical. However, the collected measurements are less rich in information than with multicast. Meanwhile, the shared links between the probing paths introduce correlations that help in the problem resolution but it would not be better than duplicating packets and having a detailed trace as in multicast. The main advantage of applying unicast in network tomography is the ability to deploy customized probing paths between the monitors without having to follow a tree structure. For both link loss rates or delay distributions, the inference problem with unicast measurement can be modeled by a linear equation system as described in (2.1) since both metrics are additive. For the loss rate, additivity holds for 1-log(transmission rate) instead, plus independence assumptions.

Paper [66] introduces a probing technique based on unicast striped packets (the delay between the sent packets is very small) to get closer to multicast functioning. Then, to infer the losses on individual links, the authors apply an estimator already proposed for multicast probing. When the measurements are perfectly correlated, i.e. most of the packets passing through a common link are dropped or passed together, the estimator is unbiased. In reality, it is very difficult to ensure the perfection of these correlations, which introduces a bias. In this case, the authors propose to use a larger number of packets to improve the estimations quality.

Unicast probing can be applied also for link delay distribution. In [6], the authors formulate this problem as a maximum likelihood estimates and solve it with an EM algorithm under the assumption of delay distribution stationarity in the observed period.

2.3.2.3 Origin-Destination traffic intensity inference

The estimation of origin-destination (OD) traffic intensity is another application of network tomography. The objective becomes the approximation of path traffic volumes from limited individual link traffic counters. The obtained OD traffic matrix is an essential input for IP routing protocols like OSPF. The link level traffic statistics are available through monitoring protocols such as SNMP, Netflow, or Sflow. Then, the estimation of the OD traffic matrix can be modeled by a linear inverse problem. The solution proposed in [67] assumes that the OD traffic flows has an independent identically distributed Poisson model and uses a Maximum Likelihood (ML) approach to estimate it. Paper [7] assumes the same Poisson model, but a bayesian approach is adopted as the inference method.

In [68], the authors propose a parsimonious traffic model for traffic matrix inference, i.e. a model described with a small number of parameters. The tests with operational network data show a good accuracy compared to the classic approaches based on the Gravity method. Also, the proposed model allows the design of traffic flows anomaly detection algorithms characterized by robust optimality properties compared to state of the art solutions [69].

2.3.2.4 Probing strategy design

Multiple statistical works in network tomography propose estimators for local metrics inference. However, the design of probing schemes is not sufficiently explored. In [8], the authors address this issue following an *experimental design* approach [70]. This method has been also used for network monitoring to find the appropriate sampling rate and the appropriate flow sampling size [71]. The proposed solution is based on the Fisher Information Matrix (FIM) that gives a measure of the information afforded by the probed paths about the link parameters. Based on the FIM information, the framework finds the best allocation of the probes among the available paths enabling to minimize the estimations error. Two illustrative applications have been tested: the loss rate and the packet delay variation (i.e. the jitter).

2.3.3 Algebraic network tomography

Statistical tomography solutions often provide estimators to approximate the unknown hidden metrics. Prior information about the unknown metrics can be added to improve the estimations. However, a unique identification of the hidden metrics remains difficult without having enough path measurements under specific conditions. As a trivial example, if two links always appear together in the supervised paths, we can't distinguish between their states. Thus, most of the recent tomography works focus on the topological conditions of the supervised networks and the probing schemes enabling the accurate identification of most of the hidden parameters.

In algebraic tomography, the link/node metrics are considered as constant unknown variables. The end-to-end measurements represent deterministic values. The proposed solutions focus on the necessary conditions on the probing paths allowing to determine uniquely the node/link parameters. For example, for additive metrics such as delay or jitter, the number of probed paths must be equal to the number of unknown variables and they must be linearly independent [9]. Generally, these constraints, expressed in terms of algebraic conditions, are often difficult to exploit in the design of monitoring strategies. Therefore, we always try to transform them in terms of graph properties, which are easier to interpret and manipulate in network problems. The majority of these solutions focuses on additive and Boolean metrics, where the inverse problem consists of solving a linear or a Boolean equation system respectively. These methods guarantee the identifiability of all the unknown metrics, but several conditions and constraints on the collection points and the probing paths must be respected.

2.3.3.1 Additive metrics

The notion of "identifiability" is introduced in network tomography. A metric on a link/node is identifiable if it can be determined from the end-to-end measurements, which implies some constraints on the supervised paths. The identifiability of additive metrics is related to the number of linearly independent paths that must be equal to the number of unknown variables to form a solvable linear equation system. So, the monitors' placement and the deployed paths between them should satisfy these constraints.

Multiple works studied the monitors' placement problem under different probing schemes. For example, in [63], the authors establishednecessary and sufficient conditions to identify an additive metric under controllable cycle-free probing paths. In this context, "controllable" refers to the ability to deploy any probing path between two monitors. This assumption becomes more realistic in new generation networks with the support of programmable equipment and SDN. The "cycle-free" assumption refers to the incapacity to deploy probing cycles starting and ending at the same monitor. These conditions are expressed as graph connectivity properties which are more practical and easier to check than linear algebra-based ones (i.e. with a linear equation system formulation). The reached conclusions show that if the topology is well connected, we need fewer monitors to ensure the total identifiability since these varied connections provide multiple choices for the probing paths deployment. Finally, they propose monitor placement algorithms that determine the minimum needed number of monitors and their placement. They are essentially based on graph decomposition techniques into connected components.

Paper [9] addresses a complementary problem. Given a topology with the monitors' locations respecting the identifiability conditions defined by [63], the objective is to establish the minimum number of paths necessary to infer the link metrics. The proposed algorithm constructs the cycle-free probing paths with a linear complexity computing time. Then, a second

algorithm solves the linear equation system with a linear computing complexity.

The identifiability of all links in a given topology usually requires a large number of monitors especially when the network graph is not highly connected. Indeed, there was also an interest on the partial identifiability of hidden parameters. For example, in [72], the proposed solution maximizes the number of identifiable links with a limited set of monitors. The proposed heuristic algorithm places the monitors incrementally while maximizing the number of identifiable link parameters. It has a polynomial computing complexity and proved to be optimal when the network topology graph is 2-vertex-connected.

The application of these techniques in dynamic environments like virtual networks adds multiple challenges and constraints. The dynamic management of virtual networks requires frequent re-configurations of the devices such as adding or removing nodes and links, moving or re-routing the network flow paths for load balancing. Thus, the monitoring strategy should take into consideration the dynamic aspect of new generation networks. Otherwise the identifiability conditions cannot be maintained along the frequent changes. In [73], the monitors' placement problem is studied in the context of dynamic networks. The objective is to find a robust minimum monitors' placement to identify an additive metric from the performed cycle-free end-to-end measurements. The proposed algorithm selects first the monitors placement for a given topology and its potential predicted changes. Then, it can be updated in case of unpredictable changes. The solution is essentially based on graph decomposition into connected components and handling the node/link removes.

In the already presented solutions, it is always assumed that a probing path cannot be a cycle. However, recent routing mechanisms such as source routing and equipment programmability through SDN make the use of loops possible. Indeed, some tomography works studied the identifiability properties of additive link metrics with combined regular and cycle probing paths. For example, in [74], the authors show that it is possible to identify an additive metric in a 3-edged connected topology with only one monitor. When this condition is not satisfied, they give a method to select the needed minimum set of monitors.

Paper [75] introduces the "link rank" topology parameter, which represents the maximum number of linearly independent paths and cycles that can be established in a given topology. This parameter can be computed linearly and the needed set of paths and cycles can be determined in a polynomial time.

Tomography solutions usually have the target of estimating metrics on hidden links or nodes from path measurements. An interesting variation of this problem consists in estimating end-to-end performance on paths, where it is impossible to make direct measurements, from other available paths. The estimation of metrics on a subset of paths is always less constrained than ensuring the identifiability of all the link metrics. This technique can be used to estimate end-to-end performance on critical and inaccessible measurement paths. Paper [76] focuses on the

monitors' placement problem to infer an end-to-end additive metric on a given set of paths. Several scenarios are discussed. For example, if we have the location of the monitors already fixed, the question is what are the identifiable paths, or what is the minimum number of monitors needed and their location to identify a given set of paths.

2.3.3.2 Boolean metrics

The supervision of some network systems do not always require the exact identification of QoS metrics such as the delay or the loss rate. Sometimes, we simply need to know the state of a service or an equipment. In this case, modeling the problem with a Boolean equation system becomes more convenient because each unknown parameter will have only two states, 'up' or 'down'. Hence, Boolean network tomography aims to infer the binary state of unknown parameters from end-to-end Boolean metrics.

The detection of failed links/nodes in the given topology can be considered as a particular case of the well-known problem of *combinatorial group testing* [77]. Briefly, the objective is to detect the failed or failing elements in a global set by testing smaller subsets. Each test indicates if there is a defecting element in the subset or not [78]. In network tomography, the subsets are represented by the probing paths. The particularity here is that the selection of these paths is constrained by the topology, the applied probing technique, and the monitors' placement. *Graph-Constrained Group Testing* [79] is a variety of group testing problems that considers the constraints imposed by specific relations between the elements in the set represented by a graph and can be applied in Boolean network tomography.

In [10], the authors introduce the application of Boolean network tomography for node failure localization in general network topologies. They first define the "maximal identifiability" metric to measure the maximum number of simultaneously detectable failed nodes in a given topology. Besides, they propose necessary and sufficient conditions to localize a fixed number of failures under different probing schemes. Paper [80] studied a similar problem where the objective is to ensure the identifiability of only a subset of critical nodes from the global topology. In [81] the proposed solution gives some insights for the design of monitoring strategies that maximize the number of identifiable nodes under different probing scenarios. Paper [82] studied the impact of topological properties on the maximal identifiability. Indeed, they establish a relation between the different typology classes (directed, undirected, trees, d-dimensional grids, and bounded-degree graphs) and the identifiability metric. Paper [83] focuses on the identifiability of failed links with probing paths or cycles. The derived results show that it is possible to identify (up to) k failed links in a given network topology if its representative graph is at least k-edge connected. These conditions are used to establish the monitors' placement and the probing strategies.

2.3.4 Topology discovery

The inference of hidden network topologies from end-to-end measurements is another interesting application of network tomography. The knowledge of the topology and the routing paths is essential for multiple operations like load balancing, resource allocation, and service placement. Meanwhile, this information is very difficult to obtain in multi-provider networks. The inference of a simple view of the topology is possible through end-to-end measurements made at the network periphery. This image, even if it is incomplete, allows a better understanding of some events such as network bottlenecks. The first works in this context focused on estimating the structure of multicast trees [84]. The mechanism used by the multicast protocol, where the original packet is duplicated at each intermediate node, introduces correlations between the measurements collected at the individual receivers. Then, statistical models are proposed to estimate the simplest tree that can explain better the collected data and the location of bottlenecks.

Later, more interest has been accorded to unicast probing. For example, paper [85] proposes a framework based on additive metrics like the delay to infer the routing topology tree from a source to multiple destinations. These measurements can be easily obtained with a traceroute tool or unicast probing. Paper [86] introduces the “sandwich” probing technique based on delay measurements for tree topology inference. The main idea is to generate three unicast packet probes, two small packets separated by a larger one. The two small packets are sent to one receiver and the larger one to a different node. As a result, the differential delay between the two small packets at the receiver is essentially introduced by the shared links with the second receiver path. Afterward, the inference of the topology from these measurements is formulated as a Maximum Likelihood Estimator (MLE). The advantage of using delay difference is to avoid clock synchronization between the different nodes.

Only some few works consider general graph topologies inference. In [87], the authors studied the estimation of the topology structure from end-to-end measurement performed between nodes located at the network edge. The solution is based essentially on two types of metrics: Path Sharing Metrics (PSMs) and Distance Metrics (DMs). The PSMs between two paths provide an approximate value of the number of links between two nodes and can be estimated from the delay or the packet loss measurements. DMs measure the distance between two nodes and can be obtained from TTL in the IP header. The main objective is to infer the simplest topology that can explain the collected data.

Topology inference can have multiple applications in the context of virtual infrastructures such as NFV networks, where the access to all the infrastructure information delivered by the multiple vendors is not trivial. Recently, some work has begun to study the application of these inference techniques in modern NFV networks. Paper [88] studied the inference of the internal structure of an overlay NFV network hosting different service chain applications from end-to-

end additive metrics. Then the problem is solved with an integer linear convex program, where the goal is to find the smallest logical topology that fits with the observations.

2.3.5 Network tomography and virtual networks monitoring

In the network slicing concept, we have different levels of resource abstraction. Each layer has only an abstracted view of the used resources (physical infrastructure, multi-domain environment, etc). To manage efficiently the shared resources, we need efficient monitoring tools that allow the identification of the causes of resource shortage. We try to detect the failure points in the underlying infrastructure due to the overbooking or a misallocation of the resources. A failure point in the resource layer causes problems in the different virtual networks that share these resources.

Network tomography solutions suit well with the flexibility of SDN and the global overview of the network which is one of the inherent features of this technology. In addition, the virtual resources paradigm is also an opportunity to enhance the scalability of the monitoring system. The virtualized technologies allow us to deploy scalable and extensible solutions for storing and processing the collected information and avoid deploying physical equipment for traffic analysis and data collection.

The principle of tomography remains applicable to new network infrastructures, but the proposed solutions often need to be adapted to take into account the particularities of the new use cases. As the best of our knowledge, only few recently published works have started to study the application of tomography in virtualized networks such as the solution proposed in [88].

2.3.5.1 Slice monitoring example

Network slicing allows creating multiple virtual networks on top of a shared infrastructure. Hence, a failed node in the shared resources impacts the slices that share it. The customer who manages the slice makes regular checks on his deployed services in the virtual infrastructure to verify if the slice provider respects the Service Level Agreement (SLA). The monitoring agent in the slice periodically reports about the measurement performed at the virtual layer to a centralized monitoring unit. This entity can detect SLA violations and transfer this information to the WAN Infrastructure Manager (WIM). The WIM manages the infrastructure where the slices are deployed. It aggregates the information transferred from the different clients and looks for the root causes of the reported incidents thanks to tomography algorithms.

Fig. 2.12 illustrates the steps and the call flow between the monitoring units in the different layers to find the potential causes of an observed SLA violation on different slices:

— A monitoring agent in each slice performs regular observations on the deployed virtual links.

— Based on the load and performance measurements of data collected from each agent and its resources verification rules, the orchestrator detects the potential shortages of the virtualized resources and reports them to the WAN Infrastructure Manager (WIM). In the example of Fig. 2.12, the monitoring system notices an SLA violation in slices 1 and 2.

— The WIM maps the virtual resources with their corresponding representations in the shared infrastructure. Thanks to the tomography algorithm, the WIM infers the possible root causes to the reported incidents and consequently, it performs remediation actions such as relocation of existing paths.

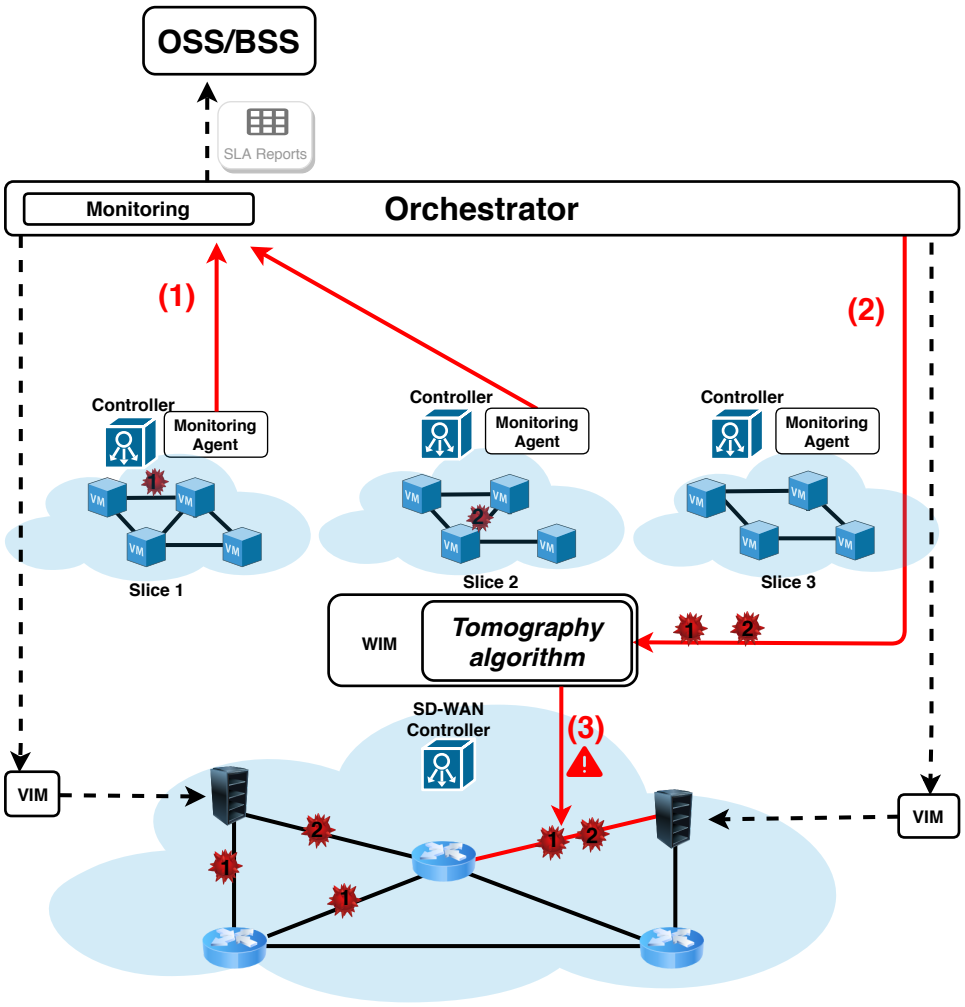


Figure 2.12 – Slices monitoring

2.4 Conclusion

In this chapter, we briefly introduce the principals of 5G, its applications and its main key enablers. We essentially focus on the emerging monitoring issues related to this new network generation. We study the existing monitoring approaches and their application in the context of NFV networks, in particular the use of end-to-end measurements to infer metrics' values inside the architecture known as network tomography. This monitoring approach allows to avoid the limitations due to needed authorization access between the different domains composing the networks.

The next chapters focus on the applicability of network tomography in the context of network slicing.

CHAPTER 3

ADDITIVE METRICS INFERENCE IN GENERAL NETWORK TOPOLOGIES

3.1 Introduction

The direct sampling of link-level metrics over large-scale IP networks, using legacy, device-centric and agent-based monitors like SNMP [31], is usually costly and cumbersome. This explains why, in practice, device-centric monitoring solutions are mostly used for static and small IP systems, such as LAN or enterprise networks. The use of network tomography [2] to infer link level performances from end-to-end observations can be a promising solution. These techniques give a global overview of the network state without collecting link level metrics from all the equipment. Another motivation for using end-to-end measurements is to provide an accurate view of the network state without consuming important network resources such as bandwidth, CPU and memory.

Unicast probing can be advantageous for such monitoring solutions due to the wide support of unicast and the easy deployment of unicast probing paths [89]. In this chapter, we introduce two statistical generic methods for the inference of additive metrics (delay or loss rate at logarithmic scale) using unicast paths. In the first proposal, called k*-paths*, we adapt the solution proposed in [90] based on the Expectation-Maximization (EM) algorithm to general topologies. The second proposal, called Evolutionary Sampling Algorithm (ESA), is a technique based on random sampling designed to avoid the high computation and memory complexity of the EM-based solutions, while keeping a similar (and good) accuracy in the evaluations. The adaptation proposed in k*-paths* gives us the possibility to compare our principal proposal ESA with the previously proposed procedures of the literature. After evaluating on some examples and discussing ESA, we proceed to a more in deep analysis of the impact of its different parameters, that allows us to propose and improvment of the technique from the performance viewpoint (similar accuracy but a significant reduction in computing time) that we called F-ESA.

The remainder of the chapter is organized as follows. Section 3.2 overviews existing works related to network tomography for the inference of additive metrics. Sections 3.3 and 3.4 present respectively the problem formulation and the proposed solutions. We study and com-

pare their performances in Section 3.5. In Section 3.6, we make a detailed analysis of the ESA algorithm parameters to find an appropriate fine tuning for each network topology. Then, we propose a faster version called the F-ESA interesting in case or large topologies) in Section 3.7. Finally, Section 3.9 concludes the work and outlines some perspectives for future developments.

3.2 Related work

Statistical inference approaches examine a set of observed measures in order to perform a mapping from path-level to link-level metrics. Generally, this is formulated as a likelihood function that needs to be maximized. These solutions may be classified according to the probing schemes used.

3.2.1 Metrics inference with multicast probing

Authors in [91] propose a solution to infer loss rates in a network topology tree using multicast probing. Basically, a designated root node actively sends multicast probes to all leaves where the monitoring traffic is collected. While this method has a good performance in terms of accuracy and convergence, it requires to satisfy multiple constraints for its deployment. The support of multicast in the intermediate nodes and the needed number of hosts in the leaves to collect detailed packet traces are its two main drawbacks in practice.

Recently, more attention has been accorded to the applicability of multicast probing with general network topologies. In [92] and [93], the authors propose to use overlapping multicast trees to cover the targeted network area. Different Maximum Likelihood Estimators (MLE) have been proposed to infer the loss rates in the intermediate links from the measurements performed over the multicast trees. These solutions have the same limitations as [91].

3.2.2 Metrics inference with unicast measurements

Less restrictive and lightweight unicast-based solutions are proposed in [94]. Basically, a server examines the native feedback of established TCP connections with a set of clients in order to passively infer observed metrics at each client side. Using that data and knowing the delivery topology, the authors designed three inference algorithms to identify the lossy links. In spite of its easy deployment, such a solution is unable to infer link-level metrics with satisfactory precision. As such, it may be only used as a preliminary support tool for network diagnostic, as explained by the authors.

In [90], a probing framework called *Flexicast* for link delay inference in a tree is proposed. Delays (the target) are discretized and a mixed method of probing between unicast and mul-

ticast is designed. The inference problem is formulated as the maximization of a likelihood function with latent variables. The maximization technique used is the EM algorithm. Paper [95] deals with the same issue and proposes an optimized implementation of the EM algorithm. In fact, it proposes to memorize some redundant operations in order to reduce the computing complexity. A fast method to detect the worst performing links in a network topology tree is described in [96] with a likelihood inference procedure. In [89], the authors propose a link-level inferring solution that identifies the maximal multicast probing spanning tree over a given general network topology. Therefore, multicast probing trees are created where conventional multicast MLE can be applied. The remaining links are covered with unicast probing paths.

3.3 General context and problem description

3.3.1 General context

Typically, a monitoring tomography solution can have three main parts: choosing the points for traffic data collection, deploying a probing strategy and inferring the internal network state. The best method to evaluate the efficiency of the first and the second steps is studying their impact on the quality of the estimations. Then, the monitoring strategy can be adapted regularly according to its efficiency and to the network topology updates. Thus, in this work, we concentrate on the inference part since it is the key of internet tomography solutions. We consider the list of probed paths as known parameters. Once the monitoring data is collected, our solution tries to infer the network state. A complete strategy for the probing scheme deployment is in the scope of our future works.

3.3.2 Network model and notation

The network topology is modeled by a directed or undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ (in some cases it can be a multigraph), where $\mathcal{V}$ is the set of network vertices and $\mathcal{L}$ is the set of network links. Let us denote by $\mathcal{P}$ the set of given probed paths. The cardinalities of sets $\mathcal{V}$, $\mathcal{L}$, $\mathcal{P}$ are respectively V, L, P, and we denote these sets $\mathcal{V} = \{1, 2, \ldots, V\}$, $\mathcal{L} = \{1, 2, \ldots, L\}$, $\mathcal{P} = \{1, 2, \ldots, P\}$. Each path π is represented by a vector of size L, where $\pi(\ell)$ is equal to 1 if link ℓ belongs to π, and to 0 otherwise. A is a matrix with rows composed of the probed paths vectors (dimensions $P \times L$). Thus, $A(p, \ell)$ is equal to 1 if path p uses link ℓ, and to 0 otherwise.

Y denotes the vector of the observed path metrics. Hence, its size is P and its pth element $Y(p)$ is the metric value on path p. X denotes the vector of the unknown link metrics. Its size is L and its ℓth element $X(\ell)$ is the metric value on link ℓ.

For efficiency reasons, we discretize the set of values that the additive considered metric can take, and we assume known an upper bound of those values. Referring to the target metric

on network links, we denote by B this bound, and we consider that the set of possible values of the metric is $\{0, \Delta, 2\Delta, \ldots, J\Delta = B\}$ for some chosen unit Δ.

In our approach, X is a random variable with a discrete probability distribution α, seen as an $L \times J$ matrix. Hence, $\alpha(\ell, j)$ represents the probability of observing value $j\Delta$ on link ℓ. Our objective is then to get an accurate approximation of the probability distribution α. Observe that the elements of every row of matrix α sum up to 1.

3.3.3 The linear model

Recall that the objective is to estimate the link metrics from end-to-end measurements. The relation between path and link level metrics is given by the following linear equation:

$$AX = Y. \tag{3.1}$$

This linear system is, in the practical cases, undetermined (that is, $P < L$ –and often $P \ll L$). In these cases, given Y, there is an infinite number of vectors X satisfying (5.1). The point here is that we have more information available, we know the network topology leading to matrix A, that makes that the random components of vector X will exhibit, in general, inter-dependencies. This information can be used to find an appropriate approximation to solutions to the previous system of equations. This can be achieved by means of statistical methods, that can then provide good approximations of the distribution of vector X, for instance maximizing the likelihood of the observed data.

On additive metrics. The additivity of the delays is easy to support. The delay experienced by a packet on a path is the sum of the delays of the packet at each link of the path. The delays are not independent of each other, but the mean delay on the path is the sum of the mean delays on the links, whatever the form taken by the dependence between link delays. For losses things are different. Consider a tandem of n finite capacity queues in equilibrium; denote by a_i the probability that a generic packet is accepted at node i and by b_i the loss probability at i. Because of the fact that packets belonging to the same flow are usually of similar sizes, that the flow which our generic packet belongs to is not alone in the network, that is, that there are many other flows sharing some of the path links and mixed with each other, and considering that the link metrics are measured averaging over many different packets, it is usually assumed that the probability that a packet can reach the end of the path is $a_1 \cdots a_n$, and that this assumption provides good approximations (see for instance [97] or [98], and the references therein). Taking logs we then have that the log of the acceptance probability can be reasonably assumed to be an additive metric.

3.4 Tomography algorithms for general network topologies

3.4.1 Introduction

This section introduces our network tomography solutions for general network topologies using unicast probing. We start by the *k-paths* method, that as previously stated, is a direct extension of the proposal in [90]. In this paper it also serves to provide performance indications about our next ESA algorithm.

In [90], the authors propose a solution for additive metrics inference in tree topologies using unicast probing, in the context of finding X in (5.1). The measurements are performed between the root of the tree and the leaves. The EM algorithm is used to find the most likely solution that matches the system. The principle of this approach is to divide the linear system into multiple smaller sub-systems. Each sub-system is equivalent to a sub-tree. The division is managed to have correlated information based on common links, available at the leaves of each sub-tree. The algorithm then solves each sub-tree. At the end, the EM procedure uses these solutions as inputs to solve the global system.

An immediate observation here is that switching from a tree topology to an arbitrary graph reveals other optimization problems related to the placement of the traffic collection points and the probed paths. In fact, in a tree topology, the traffic is injected at the root node and collected at the leaves. More sophisticated probing strategies are needed with general network topologies.

3.4.2 *k-paths* unicast probing schemes

In this section, the *k-paths* method is introduced as an adaptation of EM-based solutions for arbitrary graph topologies. This adaptation affects only the preliminary steps that precede the main calculation procedure for the inference of link metrics. Therefore, *k-paths* maintains the main steps of EM-based solutions and thus it keeps the same properties of convergence, computational complexity and memory consumption. This allows us to compare our main proposal with the state-of-the-art solutions based on the EM technique.

In the *k-paths* method, the main idea of [90] is followed, which is dividing the global linear system into multiple sub-systems with smaller sizes. However, we propose different ideas for splitting linear system (5.1) and for the resolution of the obtained sub-systems that are not limited here to tree topologies. Basically, we put the different paths in clusters maximizing the number of shared links between the paths inside them, and minimizing the number of links in each one of them.

For these purposes, a clustering algorithm is used based on reducing a distance (for example, Euclidian, or Hamming) inside the obtained clusters. The algorithm takes as its input the list of the probing paths vectors encoded in matrix A and tries to gather them with maximum

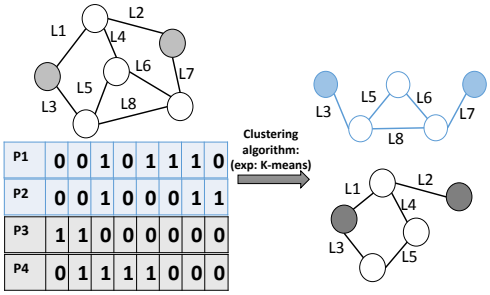


Figure 3.1 – Network topology splitting. For clarity in the picture, we denoted $L1, L2, \ldots$ the links, $P1, P2, \ldots$ the paths.

distance similarity as illustrated in Fig. 3.1. The *k-means* algorithm can be used for this task. Then, a recursive solver based on a Depth-First-Search is used to explore the solutions space and find the possible ones. Last, the EM algorithm starts from the obtained solutions and sweep them to find the best one for the global linear equation system.

Some remarks before providing the details. From a global point of view, what we want is a procedure that given a single observation Y, builds an approximated value of the link-level metrics X. In the process, from this single observation we build a potentially huge set $\mathcal{X}$ of possible vectors X (the first step), from which a first approximation of the distribution of X is computed. Then this approximation is refined by means of the EM algorithm. If this set $\mathcal{X}$ is too big, we can sample it to continue the computations with a subset of candidates having a more reasonable size. This will not be explored in deep here; the considered examples allow to deal with the space of all potential candidates.

A last remark concerning targeting α, that is, the distribution of X, and not, say, more imply the mean $E(X)$. In this paper, we illustrate the approach with the problem of approximating X from the observation of Y, but more complex applications are possible, and we mention one at the beginning of Section V where numerical results are exhibited. So, we will stay general in the description of the global approach.

3.4.2.1 Recursive solver using Depth-First-Search

The purpose of the recursive solver is to find all possible solutions for a given linear equation system taking into consideration that each variable takes integer values. A pseudo-code of the solver and the introduced variables are presented in Algorithm 1. Let us consider, for instance, the example shown in Fig. 3.2.

To further simplify things, assume that each variable can take only two values, 0 and 1. Fig. 3.3 shows the solutions space tree. Each node represents a variable and the edges issued from it are its possible values. See that there are two paths in the example, and that we have

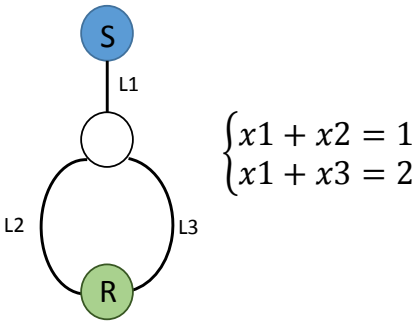


Figure 3.2 – Probing scheme example, with only two paths from sender S to receiver R.

$Y = [1, 2]$. The algorithm tries to distribute the end-to-end values among the link metrics variables using recursive calls. Thus, each recursive call has two parameters: the link index ℓ and the remainder of the end-to-end measurements Y. In a passage by variable X_ℓ corresponding to link ℓ, the link metric value, denoted j in the pseudo-code, is subtracted from all the paths p that include link ℓ ($\ell \in p$, $A(p, \ell) = 1$).

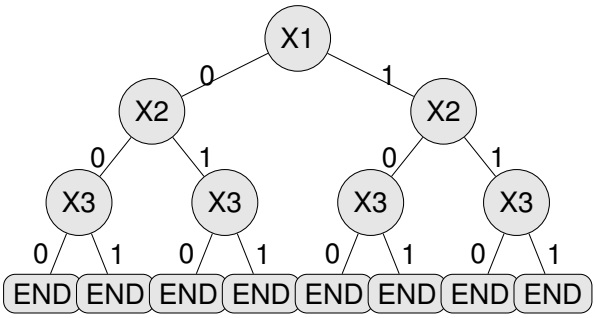


Figure 3.3 – Solution space example corresponding to Fig. 3.2. We have three variables *X1, X2* and *X3*, each taking two possible values, 0 and 1.

If the reminder on one path is strictly less than the metric value, the algorithm does not make a recursive call (line 10 in Algorithm 1). This condition avoids exploring the branches of the solution space tree that could not lead to possible solutions. When the “End“ node is reached and the remainder Y is equal to 0, the path taken by the successive recursive calls which corresponds to one solution is saved. The path made by the recursive calls is memorized in vector $\mathcal{S}$. In fact, at each recursive call, the value taken by the link metric X_ℓ is added to $\mathcal{S}$ ($\mathcal{S}$.add(j)). At the end of the call, this value is removed ($\mathcal{S}$.delete-last-value).

The first call of the solver is done from the root node X_1: *Solver*$(1, Y = [1, 2])$. In this example, the variable X_1 can take two values, 0 and 1. The algorithm makes two recursive calls:

— *Solver*$(2, Y = [1, 2])$, when X_1 is equal to 0,

— *Solver*$(2, Y = [0, 1])$, when X_1 is equal to 1.

Algorithm 1 Recursive solver

INPUT: linear equation system $AX = Y$.
OUTPUT: $\mathcal{X}$, all possible solutions to the linear system.

```
1: 𝒳 ← ∅                                          ▷ Initialize the solutions set
2: 𝒮 ← ∅                                          ▷ Vector to memorize the recursive calls
3: function SOLVER(ℓ, Y)
4:     if ℓ > L then
5:         if Y = 0 then:
6:             𝒳 ← 𝒮                               ▷ Save Solution
7:         else
8:             Return
9:         end if
10:    end if
11:    for j ∈ range(0,J) do
12:        if ∀p ∈ 𝒫, with A[p, ℓ] = 1, Y[p] ≥ jΔ then
13:            Y' = Y
14:            for p ∈ 𝒫, A(p, ℓ) = 1 do
15:                Y'[p] = Y'[p] - jΔ
16:            end for
17:            𝒮.add(jΔ)
18:            Solver(ℓ + 1, Y')
19:            𝒮.delete-last-value
20:        end if
21:    end for
22: end function
23: First call of the solver:
24: Solver(1, Y)
```

3.4.2.2 Expectation-Maximization (EM) algorithm

The proposed EM algorithm is an adapted version of the EM algorithm described in [90]. The objective is to find the probability distribution α from the probing paths matrix A and the end-to-end observations Y. To obtain an approximation of α, a matrix of latent variables M must be beforehand estimated. The number $M_{\ell,j}$ represents the expected number of times the metric takes value $j\Delta$ on link ℓ. Matrix M has the same dimensions as α.

Expectation step (E-step): The purpose of this step is computing an estimation of the latent variables matrix M from the initial values of α. For a better understanding of the problem formulation, we introduce additional notation described in Table 6.1.

Table 3.1 – **List of used parameters**

variable	description
C	The set of clusters
C^ℓ	The subset of clusters where there is at least one path that includes ℓ
y^c	Vector of end-to-end observations for the cluster of paths c
$\mathcal{X}^{c,y^c}$	The solutions to the linear equation system corresponding to c and y^c
N_{c,y^c}	The number of probing experiences where y^c is observed in cluster c

Thus, $M_{\ell,j}$ can be computed using (3.2):

$$M_{\ell,j} = \sum_{c \in C^\ell} \sum_{\substack{x \in \mathcal{X}^{c,y^c}, \\ x[\ell]=j\Delta}} \frac{\Pr(x)}{\Pr(y^c)} \times N_{c,y^c}, \tag{3.2}$$

where the probability of each solution vector $x \in \mathcal{X}^{c,y^c}$ is computed with (3.3),

$$\Pr(x) = \prod_{\ell \in c} \alpha(\ell, x[\ell]/\Delta), \tag{3.3}$$

and the probability of y^c is the addition of the probabilities of all the possible solutions as described in (3.4):

$$\Pr(y^c) = \sum_{x \in \mathcal{X}^{c,y^c}} \Pr(x). \tag{3.4}$$

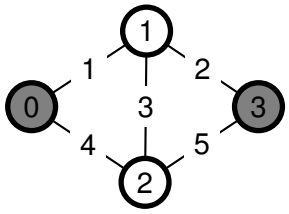
Maximization step (M-step): the new probability distribution that maximizes the likelihood of the observed data is computed from M as described by (3.5):

$$\alpha_{\ell,j} = \frac{M_{\ell,j}}{\sum_{j=0}^{J} M_{\ell,j}}. \tag{3.5}$$

The two steps are repeated until the convergence of α. More details about the used EM algorithm are available in [90].

3.4.2.3 Detailed example

In this section, we give a simple example to explain the different steps of the *k-paths* solution. Let us consider, for instance, the example shown in Fig. 3.4. It is a small topology with five links. We consider three probing paths to cover them. The associated path matrix, denoted by A is given in Fig. 3.4. To further simplify the explication, we make some assumptions. Firstly,



$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Figure 3.4 – Example

we consider only one cluster (the system is not divided into multiple ones). Secondly, we consider only one measurement vector y (N_{c,y^c}=1). Thirdly, the metric on each link takes only three values: 0, 1 and 2. Therefore, we consider that the end-to-end measurement on the three paths is represented by the vector $y = [2, 4, 2]$. The first step is finding the set of possible vectors $\mathcal{X}$ that satisfy the linear equation system by the recursive solver. The result corresponding to the given example is represented below:
$\mathcal{X} = [0\ 2\ 2\ 0\ 2; 1\ 1\ 1\ 0\ 2; 1\ 1\ 2\ 1\ 1; 2\ 0\ 0\ 0\ 2; 2\ 0\ 1\ 1\ 1; 2\ 0\ 2\ 2\ 0]$

The solution set $\mathcal{X}$ will be the input for the EM algorithm that returns the best probability distribution α for the links metrics.

Given taken assumptions and these simplified notation, the formula (3.2) can be simplified into (3.6):

$$M_{\ell,j} = \frac{1}{\Pr(y)} \sum_{\substack{x \in \mathcal{X} \\ x[\ell]=j\Delta}} \Pr(x). \tag{3.6}$$

The EM algorithm starts by the initialization of α. If there is no prior information about the network state, all the values for each metric are considered equiprobable. As a result α^0 is equal to:

$$\alpha^0 = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}.$$

Then, α^0 is used to compute P^0, the probability vector of the different solutions using (3.3). Since all the metric distribution values start with the same probability, all the solutions will also

have the same probability.

$$P^0 = [P^0(s1), P^0(s2), P^0(s3), P^0(s4), P^0(s5), P^0(s6)] \\ = [0.0041, ..., 0.0041]$$

After that, each probability is divided by the probability of y computed with (3.4). The obtained values are the scaled probabilities that help to compute M as described by (3.2). The vector of scaled probabilities is denoted P_s^0.

$$P_s^0 = P^0/Pr(y) = [0.166, ..., 0.166]$$

Then, we compute the variable M^0. For example, $M^0(1,0)$, the estimation of having value 0 on link 1, is equal to $P_s^0(1)$ since only the first solution gives the 0 value for the first link. $M^0(1,1)$ is equal to $P_s^0(2)$+$P_s^0(3)$ since the second and the third solutions give 1 for the first link.

Since we are considering a very simple example (one measurement and one cluster), α^1, has the same value as M^0, so we do not need to use (3.5) to compute it.

$$\alpha^1 = M^0 = \begin{pmatrix} 0.166 & 0.322 & 0.498 \\ 0.498 & 0.322 & 0.166 \\ 0.166 & 0.322 & 0.498 \\ 0.498 & 0.322 & 0.166 \\ 0.166 & 0.322 & 0.498 \end{pmatrix}.$$

Then, the same process is repeated with α^1. The new scaled probability vector is

$$P_s^1 = [0.087, 0.212, 0.137, 0.263, 0.212, 0.087].$$

Then α^2 is

$$\alpha^2 = M^1 = \begin{pmatrix} 0.087 & 0.349 & 0.562 \\ 0.562 & 0.349 & 0.087 \\ 0.263 & 0.424 & 0.311 \\ 0.562 & 0.349 & 0.087 \\ 0.087 & 0.349 & 0.562 \end{pmatrix}.$$

The EM algorithm properties guarantee that the likelihood of the observed data $Pr(y)$ increases after each iteration.

This process is repeated until the new value of α is approximately equal to the previous one, which is equivalent to say that the likelihood of the measurements converges.

3.4.2.4 Complexity analysis

The proposal is composed of three main steps: decomposing the global linear equation system (using k-*means*), solving the sub-systems and applying the EM algorithm. We study the complexity of each one separately.

In the first step, the k-*means* method is used to split the linear equation system. The complexity of the algorithm is $O(PL\,k\,n)$, where n is the number of iterations needed by the k-*means* algorithm and k is the number of clusters.

The second step is solving the sub-systems. The split of the linear system into multiple sub-systems enables to reduce the size of the problem by reducing the number of equations and variables. For each sub-system c, the solution space is explored using the recursive solver. The worst case occurs when all the solution space is explored, so, we have a cost in $O(J^{L_c})$, where L_c denotes the number of links in sub-system c. The complexity of this step is $O(J^{L_c}k)$, that is, exponential in L_c. Without decomposing the global equation system, the worst case complexity is $O(J^L)$.

Finally, the EM algorithm computes the probability distribution α of the global solution X. The EM algorithm consists of a number of iterations of the E-step and the M-step subroutines. In the E-step, for each subset of paths c, there are $|\mathcal{X}^{c,y^c}|$ outcomes from the solver. For each outcome we need $O(L_cL)$ multiplications to compute its probability plus additions and divisions to compute the latent variable M. For each subset c, the complexity of the E-step can be approximated by $O(|\mathcal{X}^{c,y^c}|\,L_cL)$. The M-step complexity is $\mathcal{O}(JL)$ which can be ignored in comparison with the E-step. Hence, the complexity of one iteration of the EM algorithm can be approximated by $O(|\mathcal{X}^{c,y^c}|\,L_cL\,k)$.

3.4.3 Evolutionary Sampling Algorithm (ESA)

The common idea between [90] and *k-paths* is splitting the linear equation system into multiple sub-systems and solving them. Then, the EM algorithm computes the best global solution. The last two steps (solving the sub-systems and the EM algorithm) have high memory and computation requirements. Thus, these methods are intractable with large topologies or when a fine granularity (a large J) is required.

Hence, we propose the Evolutionary Sampling Algorithm (ESA) which is a stochastic population-based method. Basically, a population is randomly chosen to verify a certain constraint. The goal is to improve the population to find the best solution within the population over iterations.

Thus, instead of solving exactly the linear equation system (or sub-systems), the basic equality is changed into an inequality with an error ε to minimize as shown in inequality (3.7):

$$||AX - Y|| < \varepsilon. \tag{3.7}$$

In (3.7), $||\cdot||$ denotes a standard matrix norm, X is a random variable and α is its probability distribution. As we will see, the objective is to find the best α that minimizes the error bound ε along the iterations. Distribution α is initialized randomly. The first step of the algorithm is searching the best α that satisfies the inequality. For this purpose, we proceed as follows. Firstly, an important number of solution vector samples (the initial population) are generated according to the initial probability density α, denoted $\mathcal{X}^{rand}$. Then, the samples that satisfy inequality (3.7) are selected; $\mathcal{X}^{\text{selected}}$ denotes the selected samples. The new probability distribution α is computed from $\mathcal{X}^{\text{selected}}$. The ratio of the selected samples $\mathcal{X}^{\text{selected}}$ should be greater than a minimum fixed threshold, noted D to ensure diversity in the selected population. Otherwise, the algorithm will converge quickly to a local optimum. In the tests, we fix D to 5%. The probability of observing value j on link ℓ, $\alpha(\ell, j)$, is computed by dividing the number of samples where link ℓ takes value $j\Delta$ by the total number of selected samples $|\mathcal{X}^{\text{selected}}|$ as described in (3.8).

$$\alpha(\ell, j) = \frac{|x \in \mathcal{X}^{selected}, x[\ell] = j\Delta|}{|\mathcal{X}^{selected}|}. \tag{3.8}$$

This process is repeated until the convergence of α or until reaching a maximum fixed number of iterations. We consider that the probability density α has converged if some standard criteria comparing previous and new α is satisfied, as in many iterative processes. We don't discuss this kind of details in this text.

The second step is to tune the error according to the result of the first step. If the probability density has converged, the tolerated error ε is reduced and the process starts again, to try a more ambitious target. Otherwise, we iterate the process with a larger error. To minimize the error, ε is multiplied by an adaptation coefficient denoted β in the pseudo-code, taking values between 0 and 1. This coefficient is set up at the initialization step before running the algorithm. The previous value of ε is memorized in a variable noted ε' at each adaptation. If α does not converge in the first step, ε is initialized again with ε'. Then, β is increased using (3.9):

$$\beta = \beta + \frac{1-\beta}{2} = \frac{1+\beta}{2}. \tag{3.9}$$

The two steps are repeated until ε stops changing (that is, until β gets very close to 1). This process is globally described in Algorithm 2 and Fig. 3.5.

Some comments on the beginning of the algorithm, when we initialize some variables. Concerning α, we can sample its values from some distribution, or we can assume the Uniform one: $\alpha(\ell, j) = (J+1)^{-1}$. Concerning ε, we first normalize X and Y by dividing them by B, and we take ε rather large (for instance, $\varepsilon = 1/2$). The coefficient β is chosen not close to 1, typically $\beta = 1/2$ as well.

Algorithm 2 Evolutionary Sampling Algorithm (ESA)

INPUT: Linear equation system $AX = Y$
OUTPUT: α

Initialize α (α^0), ε and β
iter = 0 ▷ iteration index
— Step 1:
while iter $\leq$ itermax **do**
    iter = iter +1
    Generate n random samples of solutions according to distribution α.
    Select the ones that respect $||AX - Y|| < \varepsilon$; put them in set $\mathcal{X}^{\text{selected}}$.
    If$|\mathcal{X}^{\text{selected}}|/|\mathcal{X}^{\text{rand}}| < D$, repeat Step 1.
    Compute the new α (α^{iter}) from the selected samples:
        $\alpha^{\text{iter}}(\ell, j) = \frac{|x \in \mathcal{X}^{\text{selected}},\, x[\ell]=j|}{|\mathcal{X}^{\text{selected}}|}$
    if $\alpha^{iter} \approx \alpha^{iter-1}$ **then**
        Break ▷ α has converged
    end if
end while
— Step 2:
if α has converged **then**
    $\varepsilon' = \varepsilon$
    α' = α
    ε = ε β
    iter = 0
    Repeat from Step 1
else
    β = $\beta + (1 - \beta)/2$
    $\alpha = \alpha'$
    if $\beta \approx 1$ **then**
        Return α ▷ ε has converged
    else
        $\varepsilon = \varepsilon'\beta$
        iter = 0
        Repeat from Step 1
    end if
end if

3.4.3.1 Complexity analysis

The ESA procedure is composed of two parts denoted Step 1 and Step 2. The complexity of the first step has the strongest weight. It is mainly determined by the number of random solutions $|\mathcal{X}^{\text{rand}}|$. The most costly operation is the multiplication of the matrix of random solutions $\mathcal{X}^{\text{rand}}$ by matrix A. The complexity of this operation is $O(|\mathcal{X}^{\text{rand}}|LP)$. The complexity of the

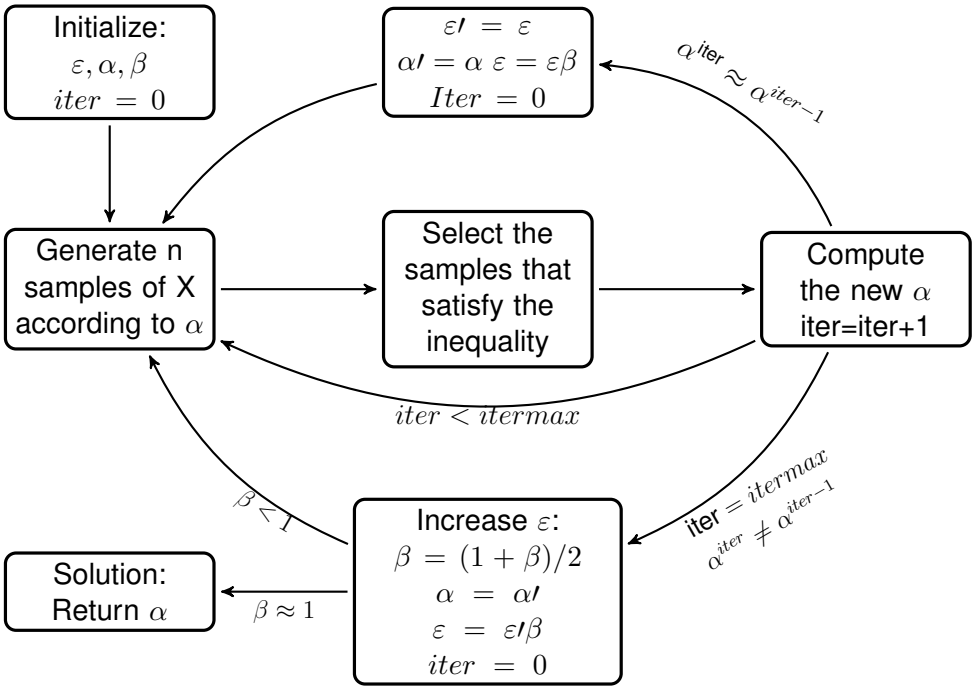


Figure 3.5 – ESA algorithm

other operations like the generation of $\mathcal{X}^{\text{rand}}$ and computing the new value of α with (3.8) depends on $|\mathcal{X}^{\text{rand}}|$ and the granularity J. Their complexity can be approximated by $O(|\mathcal{X}^{\text{rand}}|J)$. These operations are repeated until the convergence of α. Let n_1 denotes the number of iterations for the convergence of α. The two steps are repeated until the error bound ε is considered as having converged; let us denote by n_2 the number of global iterations needed. Both n_1 and n_2 are mainly influenced by the initialized parameters like the selectivity criteria ε and convergence conditions of α and β. The global complexity of ESA can then be approximated by

$$O\Big(|\mathcal{X}^{\text{rand}}|(LP+J)n_1n_2\Big).$$

3.5 Performance Analysis

The output of both *k-paths* and ESA, the two techniques proposed in this paper, is the distribution α of the unknown link metric X. This information can be useful for multiple use cases in network monitoring. One possible application is the troubleshooting of network failure using end-to-end measurement. In fact, it is possible to compute from distribution α the probability that the metric value on a link ℓ exceeds a fixed threshold $V_{\text{fail}} \in [0, B]$, where $V_{\text{fail}} = j_{\text{fail}}\Delta$,

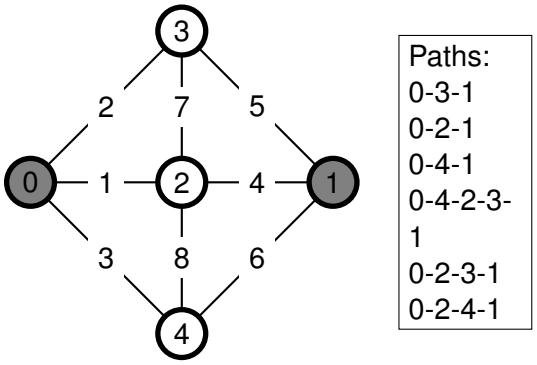


Figure 3.6 – Topology-A composed of 5 nodes and 8 links. The probing paths are established between nodes 0 and 1.

using (3.10):

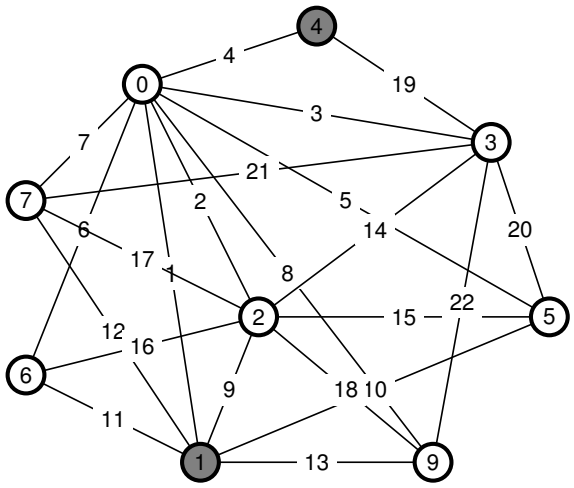
$$\Pr(X_\ell > V_{\text{fail}}) = \sum_{j=j_{\text{fail}}+1}^{J} \alpha_{\ell,j}. \tag{3.10}$$

Then, an alarm can be activated if the failure probability on a link exceeds a fixed threshold, say, if $\Pr(X_\ell > V_{\text{fail}}) > p^*$. Thus our solutions can be extended to a binary classification task for a link failure detection. The two thresholds V_{fail} and p^* can be tuned to have a small rate of false positives and false negatives in the binary classification task. This can be used for the initial troubleshooting of the network, in order to detect the potential failure root causes using only end-to-end measurements. Then, more sophisticated monitoring protocols can be applied to remote additional information about the potential failed links or equipment. Coming back to the focus of the paper, we preferred to illustrate our method with the simpler case of just approximating the value of the metric.

To evaluate the performances of the proposed solutions, we experiment on two topologies taken from [99]. The first one has five nodes and eight edges. The second graph has nine nodes and twenty-two edges. With each topology we associate a dataset of multiple samples of delay measurements performed on the different links. The traffic was simulated with the `Omnet++4` tool. For each topology, we consider a predefined list of paths and we compute the end-to-end delay on each path by the addition of the delays on the paths' links. The algorithms take as input the end-to-end delays and evaluate the estimated link metrics by computing the expectation of X from the returned α. Then, the error is evaluated using (4.3).

$$\text{Error (in \%)} = 100\frac{|V^{\text{estimated}} - V^{\text{real}}|}{B}. \tag{3.11}$$

The error and the computing time are measured according to the granularity J and the number of paths P. For a fixed number of paths, the first ones from the global list are selected.



Paths:
4-3-7-0-5-1 4-0-2-6-1 4-3-7-2-1 4-0-5-2-1 4-0-2-9-1 4-0-2-7-1
4-0-3-2-1 4-3-9-2-1 4-3-7-2-1 4-0-9-1 4-0-2-5-1 4-0-6-2-1
4-0-7-2-1 4-3-5-1 4-3-0-1 4-0-2-1 4-3-7-1 4-0-6-1 4-3-9-1
4-0-7-1 4-3-2-1 4-0-5-1

Figure 3.7 – Topology-B composed of 10 nodes and 22 links. The paths are established between nodes 1 and 4.

For each experience, we run 50 different tests.

3.5.1 Topology A

Firstly, we study the impact of the number of paths on the two algorithms. The *k-paths* solution is tested with granularity J equal to 10 (K-10) and ESA with two granularities: 10 and 50 (ESA-10 and ESA-50).

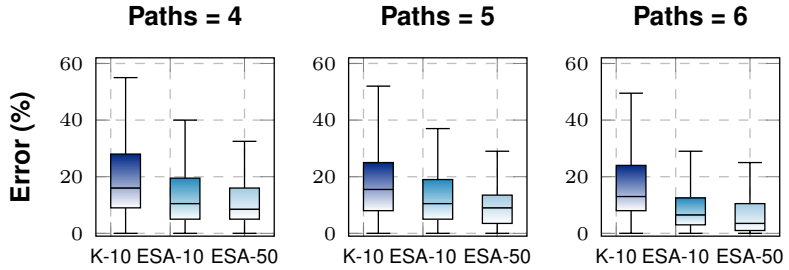


Figure 3.8 – Topology-A: Error-vs-Paths. The *k-paths* algorithm is tested with J equal to 10, while the *ESA* is run with two values, 10 and 50.

Fig. 4.4 illustrates the dispersion of the percentage error respectively of K-10, ESA-10 and ESA-50, while Fig. 3.9 illustrates the variation of the corresponding mean computing time.

For the same granularity 10, ESA-10 shows better performances than K-10. For example, with four paths, the median errors of the two algorithms are 16% and 11% respectively. The variance of the error is also enhanced with ESA-10 since the error takes values between 0 and 40% with ESA-10 and between 0 and 55% with K-10. This enhancement with ESA can be explained by the fact that it is more robust to local optima compared to *k-paths*. In fact, ESA starts by exploring the solution space with random sampling and enhances the accuracy over the successive iterations, while *k-paths* starts from one point and tries to maximize the likelihood of the observed data.

ESA-10 also shows better performances in terms of computing time. In the experience with four paths, the mean computing time with K-10 is more than 10 seconds while it is less than 1 second with ESA-10. Increasing the granularity J to 50 with ESA-50 enhances the accuracy of the estimation and increases the computing time compared to ESA-10. K-10 still have the largest computing time compared to ESA-10 and ESA-50. Increasing the number of paths enhances the accuracy of the estimations for the three experiences. This is a trivial and expected result since each path provides an additional equation for the linear system that can help in the problem resolution. Increasing the number of paths has a low impact on the computation time. In fact, increasing the number of equations in the linear system has a linear impact on the computation complexity of the worst case as explained in Sections 3.4.2.4 and 3.4.3.1.

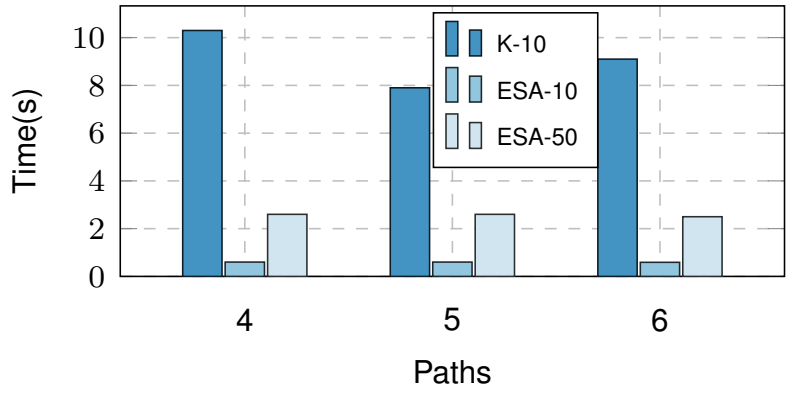


Figure 3.9 – Topology-A: Time-vs-Paths. The *k-paths* algorithm is tested with J equal to 10, while the *ESA* is run with two values, 10 and 50.

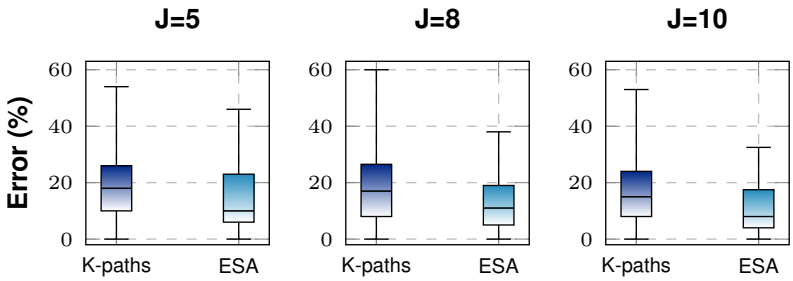


Figure 3.10 – Topology-A: Error-vs-J

However, providing additional equations can help the algorithms to converge faster, which can explain some variation of computing time values in Fig. 3.9.

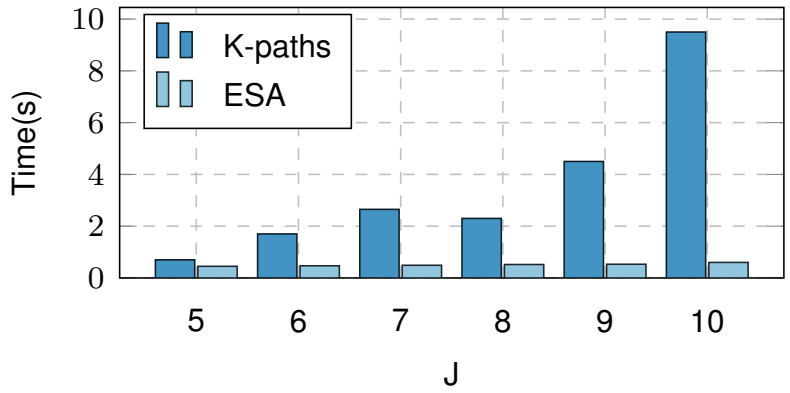


Figure 3.11 – Topology-A: Time-vs-J

Secondly, we study the impact of the granularity J. Fig. 3.10 shows the variation of the percentage error of the two algorithms as a function of J. Increasing the granularity J has an important effect on the enhancement of the accuracy of both procedures, especially of ESA. Fig. 3.10 shows that by increasing J from 5 to 10, the maximum error value decreases from 47% to 34% for ESA. However the maximum error decreases only from 55% to 52% for the *k-paths* solution. Fig. 3.11 shows the computing time variations according to J. The experimental results are in conformity with the complexity studied in Sections 3.4.2.4 and 3.4.3.1. We can observe how the computing time increases exponentially in *k-paths* and linearly with ESA.

3.5.2 Topology B

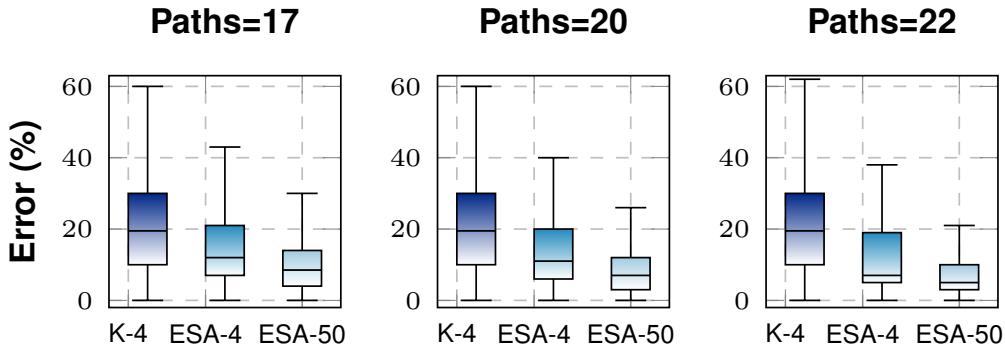


Figure 3.12 – Topology-B: Error-vs-Paths. The *k-paths* algorithm is tested with J equal to 4, while the *ESA* is run with two values, 4 and 50.

We make similar tests with a larger topology described in Fig. 3.7. Fig. 4.5 illustrates the

variation of the percentage error as a function of the number of paths. The *k-paths* procedure is tested with a granularity J equal to 4 (K-4) and ESA with two different values of J, 4 (ESA-4) and 50 (ESA-50). Fig. 3.13 shows the mean computing time of these tests.

For $J = 4$, ESA-4 has always the best results in terms of accuracy and computing time compared to K-4. At one point, adding additional paths does not enhance the estimations. In fact, there is a sampling error due to the discretization of the solution space that can be avoided only by choosing a finer granularity. With the EM-based algorithm, a higher granularity means that the computing time increases significantly (recall that the evolution is exponential). ESA is then tested with granularity 50 (ESA-50). The results show that the percentage error of ESA-50 is significantly enhanced by increasing the number of probing paths. In fact, the maximum error decreases from $35\,\%$ to $21\,\%$ and the median decreases from $10\,\%$ to $5\,\%$ with ESA-10. However, we do not have the same enhancement with ESA-4 and K-4.

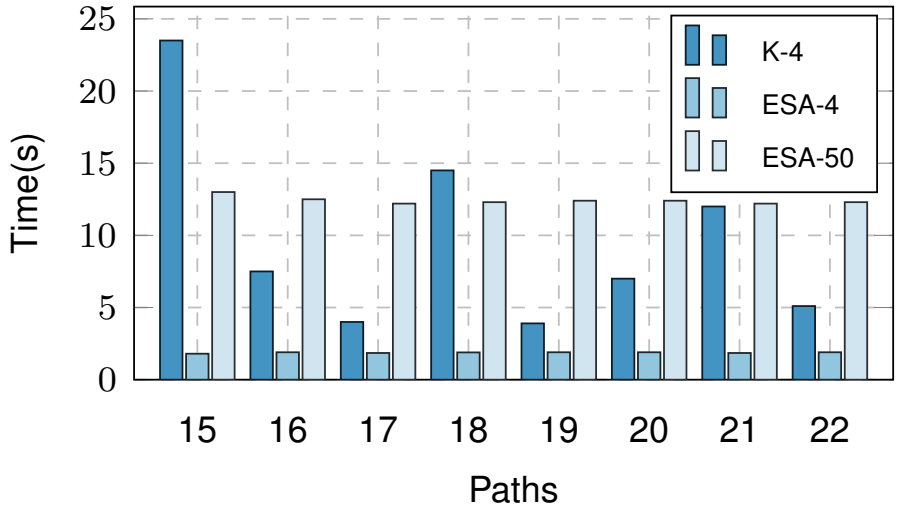


Figure 3.13 – Topology-B: Time-vs-Paths. The *k-paths* algorithm is tested with J equal to 4, while the *ESA* is run with two values, 4 and 50.

3.6 ESA fine tuning

In this section we perform a deeper analysis of ESA, that will inspire the improved version presented in the following one (Section VI).

3.6.1 Methodology

The comparison in Section 3.5 between the two proposed approaches for the inference of additive metrics, namely ESA and *k-paths*, shows that the ESA algorithm outperforms clearly

the EM-based solution. The EM-based solution is intractable with large topologies or a small granularity J. Consequently, we focus in this section only on the ESA solution to make a deeper analysis with much larger topologies. As shown in Section 3.4.3, ESA depends on multiple parameters that can be fine-tuned in order to achieve good performances with reasonable computing time.

One of the most important parameters is the number of generated samples according to α in each iteration, denoted by $|\mathcal{X}^{\text{rand}}|$. These generated samples enable to explore the solution space of the pair (J, L). On one hand, there are $J \times L$ variables corresponding to matrix α to be estimated, and the number of samples should be large enough to cover the solution space. On the other hand, the computing time complexity grows linearly with $|\mathcal{X}^{\text{rand}}|$. For this reason, a tradeoff between the number of samples and J is required to ensure good accuracy in an acceptable computing time.

The granularity J is also an important parameter for the algorithm, especially for the accuracy. Increasing J usually enhances the model accuracy but not necessarily. In some cases, increasing J while exploring the solution space with a small number of generated samples $|\mathcal{X}^{\text{rand}}|$ do not enhance the accuracy. Exploring a solution space with a small number of samples compared to its size usually leads to local optima. Therefore the two parameters J and $|\mathcal{X}^{\text{rand}}|$ should be studied together to find the most appropriate combination.

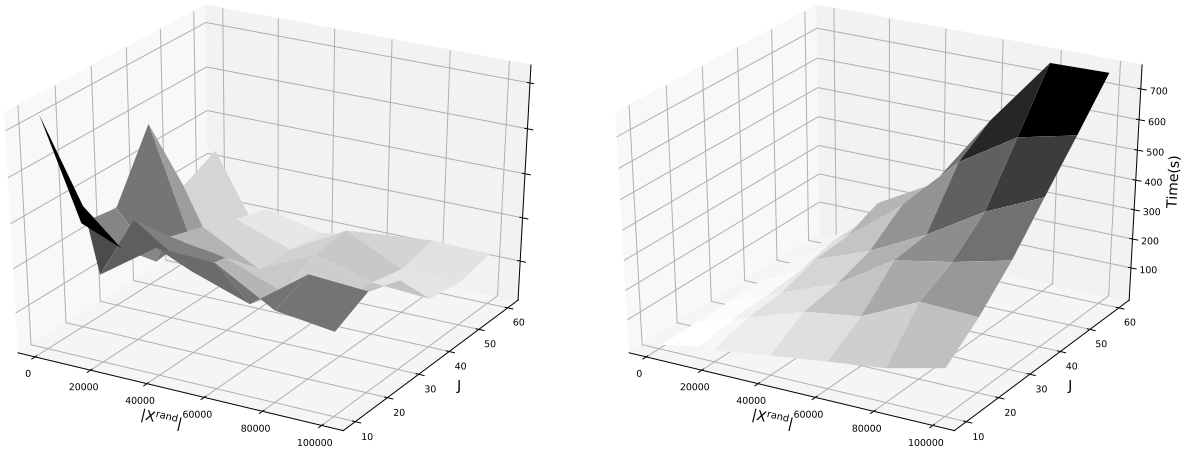
Another parameter that should be studied is the fixed threshold to consider that α has converged, denoted by τ. Regarding the computing time, increasing τ will increase the needed number of iterations for the convergence of α, which we denoted by n_1. Concerning the procedure's accuracy, increasing τ will push the algorithm to refine the estimates. By making multiple iterations, the generated samples $\mathcal{X}^{\text{rand}}$ lose their diversity which increases the risk of convergence to local optimums.

The dependence on several parameters makes it difficult to find the best algorithm parameters configuration. Here, we basically follow an experimental approach to study these different parameters and get some information about the relation between them and the global behavior of the algorithms. In order to explore these three parameters, $|\mathcal{X}^{\text{rand}}|$, J and τ, we cross in each experience two of them. We evaluate the algorithm performances (the accuracy and the computing time) for multiple pairs to obtain a 3D plot.

3.6.2 Probing path selection

We test the algorithm with two topologies from [100] and [101] with characteristics described in Table 5.1. Fig. 3.14 and Fig. 3.15 present respectively the measured performance with topologies C and D.

Table 3.2 – Network topologies



(a) Mean percentage error vs $|\mathcal{X}^{\text{rand}}|$ and J.

(b) Computing time vs $|\mathcal{X}^{\text{rand}}|$ and J

Figure 3.14 – Topology C

Toplogy	C-NewYork	D-Pioro
Vertices	16	40
Edges	51	89

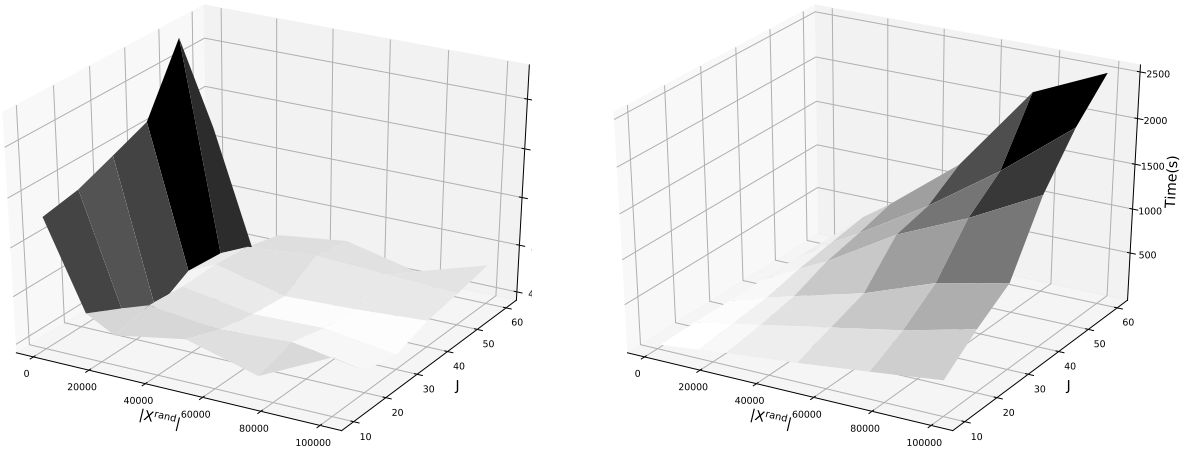
Compared to topologies A and B, the two examples studied in this section are pretty larger. Thus, we need to automate the process of the probing path generation. For this purpose, we use a strategy for probing paths deployment we just proposed in [17]. Briefly, the main idea is to cover the topology only with probing cycles and maximizing the rank of matrix A. The advantage of using probing cycles is that it allows us to avoid the synchronization issue between the sender and the receiver. For the details see [17]. Note that in this paper we focus essentially on the inference step. The probing strategy is just used to automate tests with these large test cases.

3.6.3 Results: $|\mathcal{X}^{\text{rand}}|$ vs J

In the first experience, we cross the number of samples $|\mathcal{X}^{\text{rand}}|$ with the granularity J to see their joint impact on the algorithm performances.

For the two studied examples, the number of generated samples $|\mathcal{X}^{\text{rand}}|$ is the most influencing parameter. Increasing $|\mathcal{X}^{\text{rand}}|$ usually enhances the model's accuracy. For example, as shown in Fig. 3.14 with topology C, the maximum error reaches 6.5% with 1000 samples while it is around 4.5% with 10^5 samples. The same observation can be done with the results obtained on topology D since the error attains 8% with 1000 samples and is always less than 4.5% with 10^5 samples.

Increasing J usually enhances the accuracy of the procedure and of course also increases



(a) Mean percentage error vs $|\mathcal{X}^{\text{rand}}|$ and J. (b) Computing time vs $|\mathcal{X}^{\text{rand}}|$ and J

Figure 3.15 – Topology D

the computing time. However, when the number of generated samples $|\mathcal{X}^{\text{rand}}|$ is relatively small, the estimation error presents multiple variations. Increasing J makes the solution space larger which requires a large number of samples $|\mathcal{X}^{\text{rand}}|$ to be covered. Thus, it is possible to converge to a local optimum since the solution space cannot be well explored.

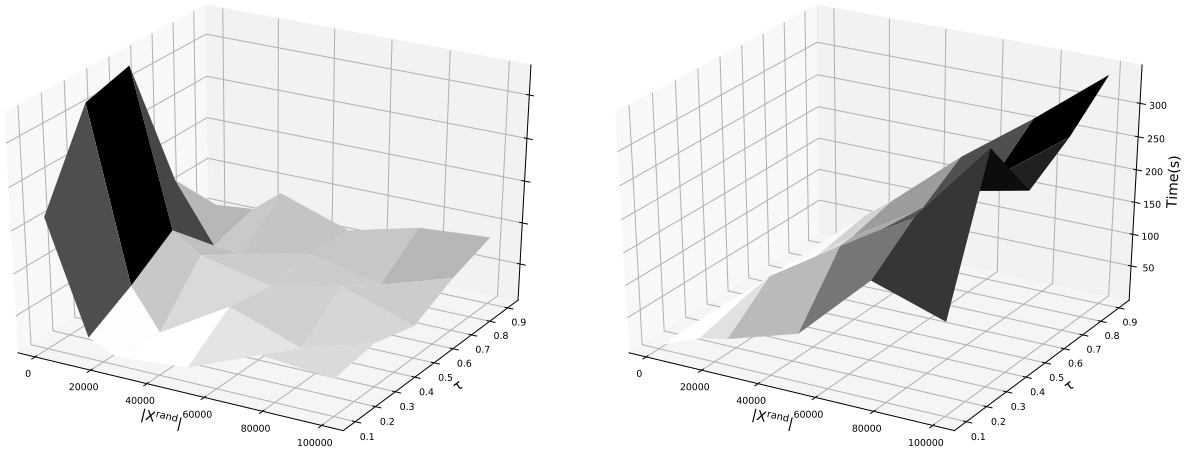
Concerning the computing time, the numbers match the results of Section 3.4.3.1. Fig. 3.14 and Fig. 3.15 show that for a fixed value of $|\mathcal{X}^{\text{rand}}|$ or J, the computing time grows linearly in the other parameter. When we increase simultaneously the two parameters analyzed here, the computing time naturally grows much faster.

3.6.4 Results: $|\mathcal{X}^{\text{rand}}|$ vs τ

In the second experience, we fix the granularity J to 20 and we explore what happens when using different values of $|\mathcal{X}^{\text{rand}}|$ and τ.

As in the first experience, the number of samples is the main influencing parameter for the model accuracy. As shown in Fig. 3.16 and Fig. 3.17, with 1000 generated samples, the estimation error reaches 10% with topology C and 9.5% for topology D. Meanwhile, with 10^5 samples, it is less than 7%. We can notice that varying τ while keeping a small value of $|\mathcal{X}^{\text{rand}}|$ does not make a considerable enhancement. Indeed, increasing τ makes the algorithm more selective with the generated examples $\mathcal{X}^{\text{rand}}$ by making multiple iterations of the first step (lines 3-12 in Algorithm 2). As a result, the generated samples $\mathcal{X}^{\text{rand}}$ lose their diversity with each iteration, which can lead to a local optimum problem.

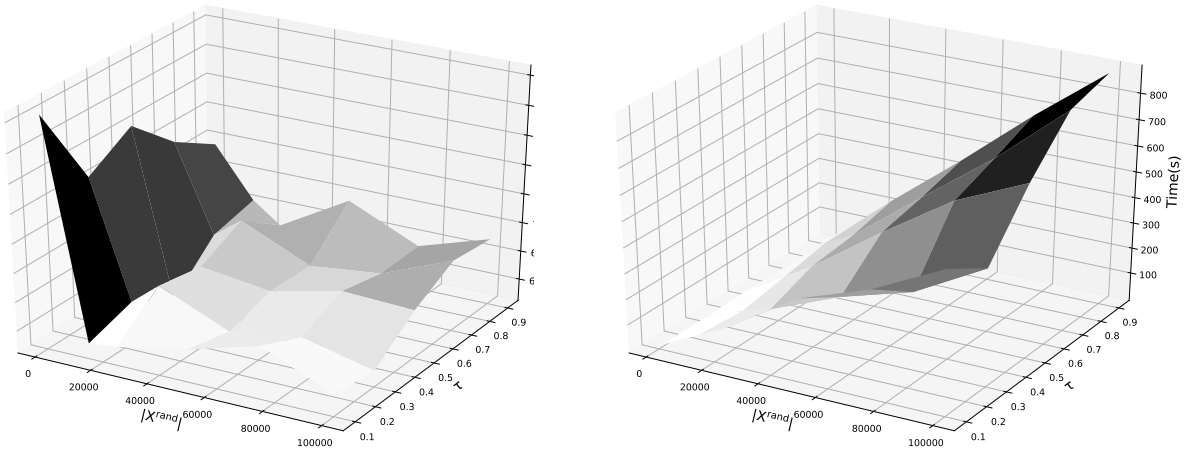
The computing time increases with τ which is an obvious result of the convergence con-



(a) Percentage error vs $|\mathcal{X}^{\text{rand}}|$ and τ

(b) Computing time vs $|\mathcal{X}^{\text{rand}}|$ and τ

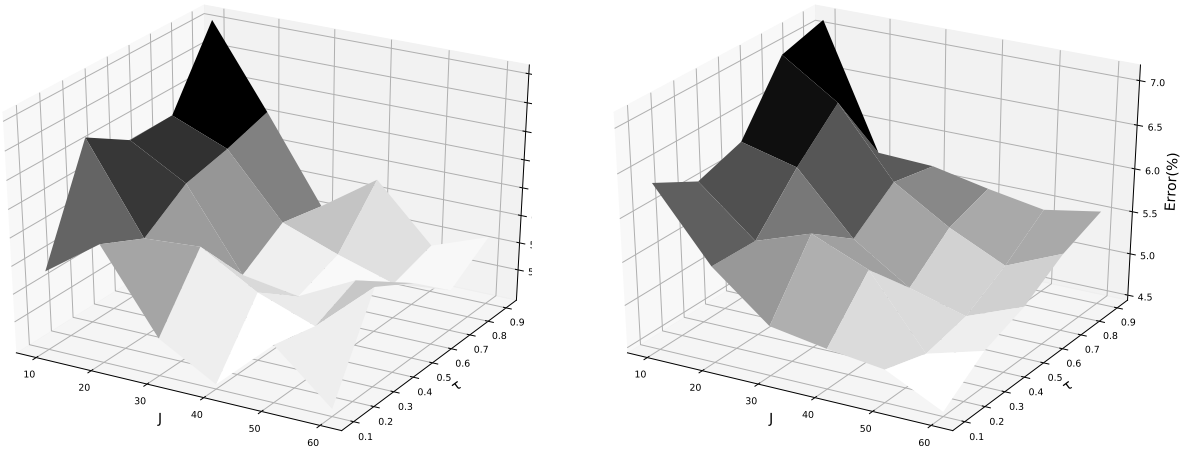
Figure 3.16 – Topology C



(a) Percentage error vs $|\mathcal{X}^{\text{rand}}|$ and τ

(b) Computing time vs $|\mathcal{X}^{\text{rand}}|$ and τ

Figure 3.17 – Topology D



(a) Percentage error vs J and τ while $|\mathcal{X}^{\text{rand}}| = 5000$ (b) Computing time vs J and τ while $|\mathcal{X}^{\text{rand}}| = 50000$

Figure 3.18 – Topology C

dition for α; it usually requires a large value of n_1. However, the plots presented in Fig. 3.16 and Fig. 3.17 show some oscillations. This can be due to a fast convergence to a local optimum, which means a small number of global iterations n_2 (see the notation in the complexity discussion at the end of Section IV).

3.6.5 Results: J vs τ

Fig. 3.18 shows the algorithm performances for different pairs of values of the parameters J and τ. We test with two values of $|\mathcal{X}^{\text{rand}}|$: 5000 and 50000.

The main observation here is that increasing J enhances the accuracy as shown in the previous experiences. The results also show that the general shape of the curve is essentially determined by J. This result, as well as other previous experiences, shows that the τ parameter is less influential on the accuracy of the algorithm than $|\mathcal{X}^{\text{rand}}|$ and J.

3.7 Fast Evolutionary Sampling Algorithm (F-ESA)

This section describes an improved version of our ESA procedure, taking into account the results of previous analysis.

3.7.1 Main idea

In spite of the significant improvement achieved by the ESA algorithm compared to the EM-based solution, the computing time increases significantly when a fine granularity is required

with large topologies. For example, for J equal to 60 and $|\mathcal{X}^{\text{rand}}|$ equal to 10^5, the computing time reaches 700 and 2500 seconds for topologies C and D respectively.

Here, we propose the Fast Evolutionary Sampling Algorithm (F-ESA) which essentially reduces the computing time needed by ESA while keeping a good accuracy in the measuring procedure. The F-ESA algorithm is an adaptive version of the ESA algorithm for large topologies.

The main idea is to divide the optimization process into two main steps: the Exploration Step where we explore the solution space with a small value of J, denoted J^{explore}, and the Fitting Step where we seek a finer granularity, denoted J^{fit}. In the Exploration Step, the objective is to move inside the solution space with $J = J^{\text{explore}}$. This enables to identify the potential zone where to look for an appropriate solution using a small computing time.

The output of the Exploration Step is the potential distribution of the link metrics, which we denote by α^{explore}, with dimensions $L \times J^{\text{explore}}$. The distribution α^{explore} will be the initialization point for the Fitting Step. However, the Fitting Step requires an initialization matrix of size $L \times J^{\text{fit}}$. Consequently, α^{explore} must be adapted for this purpose and the dimensions of α^{explore} are expanded to become a $L \times J^{\text{fit}}$ matrix.

To simplify the description but without losing generality, we consider that J^{fit} is divisible by J^{explore}. The ratio between them, $J^{\text{fit}}/J^{\text{explore}}$, is denoted R.

Then, for each j in the range $[rR, (r+1)R[$ (with $r \in [0, R-1[$), $\alpha^{\text{fit}}_{\text{init}}(i,j)$ is equal to $\alpha^{\text{explore}}(i,r)$ divided by R. Note that if J^{fit} is not divisible by J^{explore}, the ranges described above can be easily adapted to cover the range $[0, J^{\text{fit}}[$.

Finally, $\alpha^{\text{fit}}_{\text{init}}$ will be the initialization point for the Fitting Step. The algorithm will explore in-depth only the components identified in the Exploration Step as potential solutions, to accelerate the computing time. This process is described in Algorithm 3.

3.7.2 Results

In this section, we compare the Fast Evolutionary Sampling Algorithm (F-ESA) to the ESA algorithm regarding the accuracy and computing time. The tests are made with topologies C and D.

Fig. 3.19 and Fig. 3.20 present respectively the percentage error and the mean computing time for the tests made with topology C using multiple values of J.

Regarding the percentage error, the two algorithms usually have the same performances. The difference between them does not exceeds 1%, excepting in a few cases, like for J equal to 50, where the error with the ESA algorithm is 2% less than when using F-ESA. Meanwhile, the difference between them is important concerning the computing time. Fig. 3.20 shows that execution times grow much faster with the ESA solution according to J than the F-ESA while

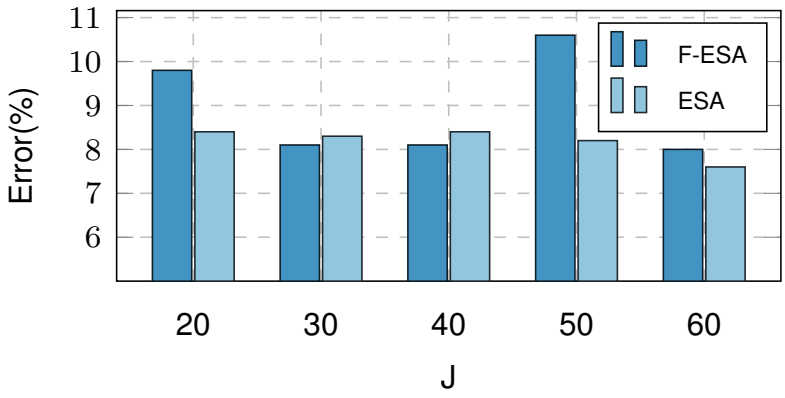


Figure 3.19 – Comparing the percentage error for F-ESA and ESA with topology C for different values of J

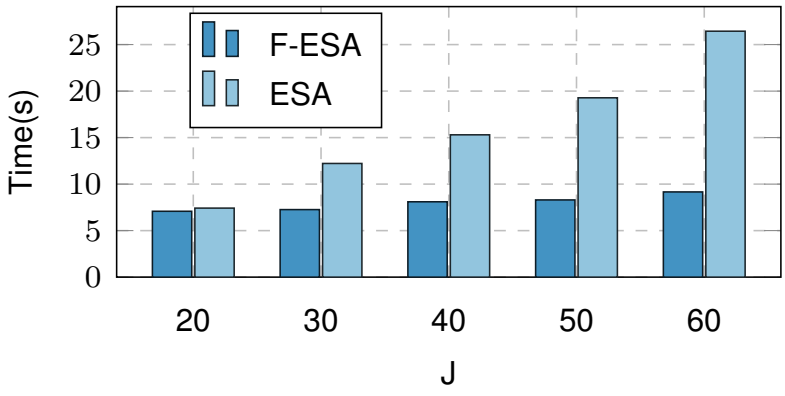


Figure 3.20 – Comparing the computing time for F-ESA and ESA with topology C for different values of J

Algorithm 3 Fast Evolutionary Sampling Algorithm (F-ESA)

INPUT: linear equation system $AX = Y$, J^{explore}, J^{fit}, τ^{explore}, τ^{fit}.
OUTPUT: α.

Exploration step:

1: $\alpha^{\text{explore}} \leftarrow \text{ESA}(\alpha_{\text{init}}^{\text{explore}}, J^{\text{explore}}, \tau^{\text{explore}})$

Expand α^{explore}:

1: **for** i $\in$ range(0,L) **do**
2: **for** j $\in$ range(0,J^{fit}) **do**
3: $\alpha_{\text{init}}^{\text{fit}}(i,j) = \alpha^{\text{explore}}(i, j/R)/R$
4: **end for**
5: **end for**

Fitting step:

1: $\alpha^{\text{fit}} \leftarrow \text{ESA}(\alpha_{\text{init}}^{\text{fit}}, J^{\text{fit}}, \tau^{\text{fit}})$
2: **return** α^{fit}

keeping approximately the same error rate. For example, with J equal to 60, the computing time is around 26 seconds with the ESA algorithm, while it is less than 9 seconds with F-ESA.

Fig. 3.21 and Fig. 3.22 illustrate the same tests made with topology D. The results confirm the previous observations concerning the model accuracy and the computing time. The ESA algorithm is always performing better than the F-ESA regarding the mean error, but the difference between them is around 1%. However, the F-ESA presents a great advantage in the computing time, especially with high values of J.

3.8 What about non-additive metrics?

In this paper we focused essentially on the inference of additive link metrics from end-to-end measurement. The proposed solutions, in particular the ESA algorithm and F-ESA, have shown good performances in the inference of this type of metrics such as delays and loss rates on a logarithmic scale. These two parameters are very good indicators of the reliability of a network system. An obvious question is whether the general principle of the ESA algorithm can be adapted to other types of metrics.

For additive metrics, the process that gives the end-to-end measurement is represented by the linear transformation given in Relation (5.1). In the case of non-additive metrics, the relation between X and Y is some other type of function f, where $Y = f(X)$, but the general

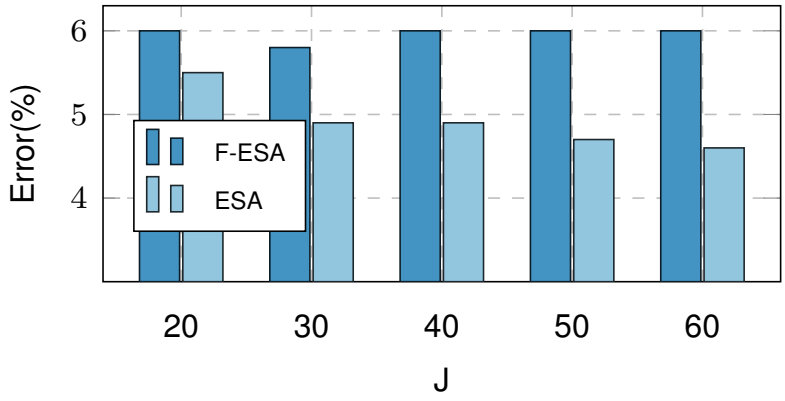


Figure 3.21 – Comparing the percentage error for F-ESA and ESA with topology C for different values of J

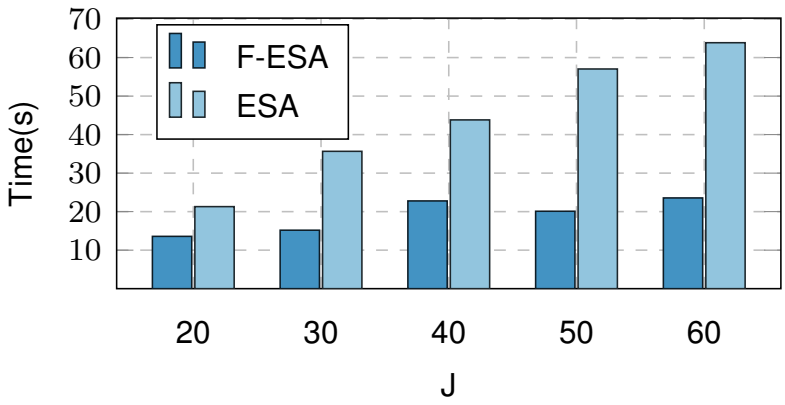


Figure 3.22 – Comparing the computing time for F-ESA and ESA with topology D for different values of J

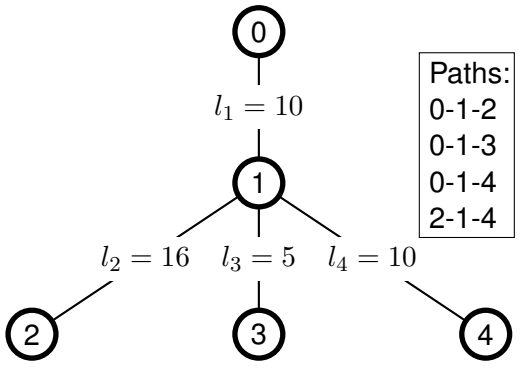


Figure 3.23 – Topology-E composed of 5 nodes and 4 links. We associate for the different links the bandwidth values 10, 16, 5 and 10 respectively. The end-to-end measurements for the probes paths are respectively 10, 5, 8 and 8.

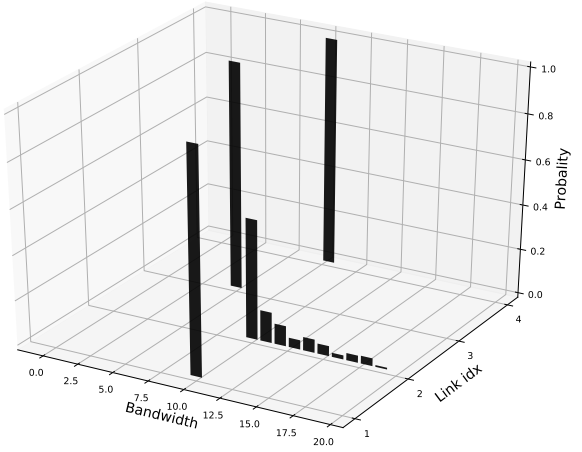


Figure 3.24 – Topology E: Probability density estimation for the available bandwidth on each link.

measuring processes described here can be adapted in order to deduce X from measuring Y. The inference step corresponds to an inverse problem, where we try to find or approximate f^{-1}. The particularity of each metric will just require some adaptation of the optimization process.

An interesting example of non-additive metrics is the available bandwidth. The available bandwidth on a path is defined as the minimum bandwidth available on the links composing it. Thus, each path metric $Y(i)$ can be computed by (3.12):

$$Y(i) = \min_{j \in [1,V] \& A(i,j)=1} X(j) \tag{3.12}$$

The objective of the inference task in this case is to estimate the closest minimum bound of each link metric to the real value. This allows the identification of bottleneck links for example. Sophisticated methods are available to evaluate links' bandwidth from end to end measures with good accuracy and without saturating the path [11]. Basically, the idea is that based on the delay and the packet size, it is possible to estimate the available bandwidth.

The procedure followed by the ESA technique described in Algorithm 2 can be easily adapted to the non linear case. The main required adaptation is to replace the linear transformation in line 7 to the computation of Y from X by the relation described in (3.12). The general principle and the rest of the process can then be used exactly as with linear metrics.

Fig. 3.23 presents a small topology where we associate an available bandwidth with each link. Then we use the adapted ESA algorithm to estimate the link metrics from the end-to-end bandwidth. Fig. 3.24 illustrates the estimated probability density for each link metric. The results show that the algorithm accurately estimates the metrics for most links, including l_1, l_3 and l_4. For link l_2, where the bandwidth is equal to 16 the algorithm gives the closest lower bound to

the true value that can be estimated.

3.9 Conclusions

In this chapter, we focus on the problem of inferring link metrics from end-to-end measurements on a network with an arbitrary topology, in the case of additive metrics and using unicast probing.

We first introduce the *k-paths* technique that extends the applicability of an existing solution hat works on tree topologies to arbitrary graphs. It is essentially based on the EM algorithm and characterized by a high computational complexity. Then, our main proposal called ESA is described. It enhances the performances of *k-paths* as illustrated in the chapter, since it gives the best results on the comparisons made on some topologies. This technique is inspired by the EM algorithm steps and the principles of genetic algorithms. It is an iterative method with multiple convergence parameters.

Despite the improvements made by the ESA algorithm compared to the EM approach, it may have some scalability problems if a fine granularity with a large topologies is required In the last part of the chapter, we make a deep analysis of the impact of the different parameters of ESA on its performance and calculation time. This leads to an improved version which reduces the computation time in a significant way. Based on a thorough study of the parameters, we propose their adaptive tuning in this last version called F-ESA (Fast ESA). The main idea is simple and it is based on the division of the optimization process into two steps: first, an exploration of the solution space with a large granularity to identify potentially interesting areas and reduce the dimensions of the solution space, and second, a refinement step using a finer granularity. The results show a significant improvement in computing time while keeping almost the same accuracy.

In the next chapter, we try to solve the same inference problem following a machine learning approach and we compare it with classical statistical methods.

CHAPTER 4

NEURAL NETWORK TOMOGRAPHY

4.1 Introduction

The use of machine learning in network monitoring is beginning to gain more importance compared to the classical statistical techniques. It has proven to be very effective in solving very complex and difficult to model problems. The use cases studied are often classic applications of machine learning techniques like the prediction of metrics on network performance, traffic variations over time, and the detection of failed links or nodes from locally collected data. In this chapter, we present a new tomography approach for link metrics inference. We model the inference task as a regression problem that we solve using a Feed Forward Neural Network model. Since getting labeled data for the training phase can be extremely costly, our procedure generates artificial data for the training phase. By creating a large set of random training examples, the Neural Network learns the relations between the path and link level metrics. This approach takes advantage of efficient Machine Learning solutions to solve a classic inference problem. Compared to the classic statistical methods presented in chapter 3, this solution does not need sophisticated methods to fine-tune the model since this is done automatically by the used learning tools. Besides, the time required for the training is very short and does not need a huge volume of data. This allows us to make fast adaptations to deal with network updates.

We can notice a lack in the literature of studies that explore the application of machine learning techniques in network tomography, although, it can be a very interesting and promising prospect. A recent work in [102] follows this direction. The authors propose to infer the end-to-end unobserved performances between different node pairs from a limited set of path measurements. The solution is based essentially on a deep neural network model that is trained to approximate the relation between the measured and the hidden metrics. The strength of neural network-based models is their ability to capture and learn non-linear relations. This property helps reducing the assumptions often considered in tomography solutions such as linearity of relationships or temporal and spatial independence.

The remainder of this chapter is organized as follows. Section 4.2 introduces the neural network model architecture. Section 4.3 gives an overview of the approach we follow based on neural networks. The testing environment and the analysis of the obtained results are presented in Section 4.4. Finally, Section 4.6 concludes the chapter.

4.2 Neural Networks models (NN)

Generally, a Neural Network (NN) model is composed of three main parts: the input layer, the hidden layers, and the output layer. Each layer is composed of multiple nodes called neurons. Fig. 4.1 shows the basic element of neural networks.

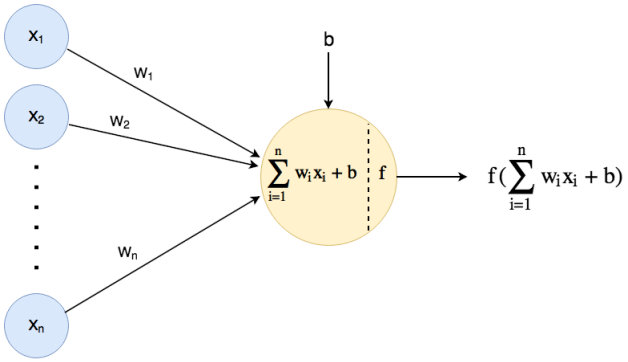


Figure 4.1 – Neuron

Each neuron has two main components: the linear part and the activation function. First, the neuron combines the different input with their corresponding coefficients/weights. The inputs are the outcomes of the previous layer. Generally, each neuron has an extra input which is the bias noted b in Fig. 4.1. The output of the linear part, which is a linear combination of the inputs, is passed through an activation function to get the final output.

The different layers of the NN architecture can be fully connected. In fact, each neuron of the hidden layers or the output layer takes as input all the outcomes of the previous layer. The dimension of the input layer is equal to the number of features of the input training data. For the output layer, its dimension is equal to the number of predicted labels.

4.3 Neural Network-based tomography

4.3.1 Network model and notation

Following the same notation given in chapter 3, the problem can be formulated by (4.1). The objective here is to evaluate X knowing Y and A, that is, to find an appropriate solution to this linear system. The typical situation here is that of an undetermined system (in practice, the number of equations P and the number of unknowns L satisfy, in general, $P < L$).

$$AX = Y. \tag{4.1}$$

The key feature of supervised learning is the ability to find complex relations between the inputs and their corresponding outputs. Multiple efficient models have been proposed in the literature for this task. We propose to model our problem as a regression task with a fully connected NN architecture as presented in Fig. 4.2.

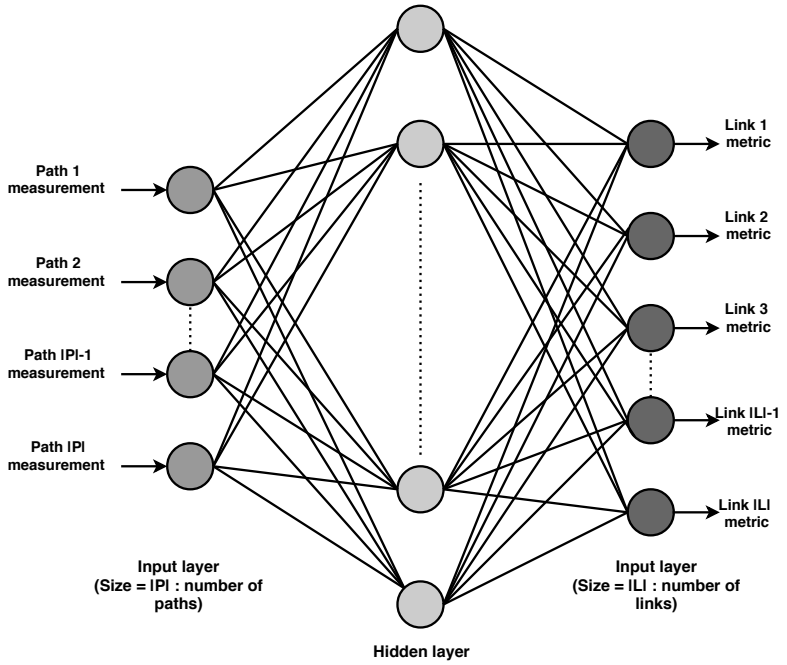


Figure 4.2 – Neural Network (NN) architecture

The dimension of the input layer is equal to the number of paths noted P, while L is the number of outcomes of the output layer. The dimension of the hidden layer is a variable parameter. In our proposal, we use only one hidden layer which is sufficient. This layer should has a large dimension compared to the input and the output layers. Thus, it gives multiple possible combinations to describe the outputs as a combinations of the inputs. At some point, increasing in the number of layers or its dimensions has no effect on reducing the error between the real values and the estimations. In our implementation, we choose the Rectified Linear Unit (ReLu) as an activation function.

4.3.2 Additive metrics inference with Neural Networks

Our proposal is to transform the inference inverse problem described by (4.1) into a regression problem and to solve it using a Neural Network. For this purpose, since we know the architecture of the network, we can simulate the distribution of traffic through the network in many configurations, and observe many pairs (X, Y). This can then be used to learn the connection between the two vectors, in the sense $Y \rightarrow X$, following a Machine Learning approach.

Figure 4.3 resumes this process.

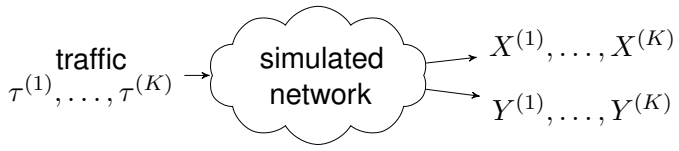


Figure 4.3 – We inject K different traffic configurations $\tau^{(1)}, \ldots, \tau^{(K)}$ into a simulated network and we measure both path metrics Y and link metrics X, obtaining K pairs $(X^{(k)}, Y^{(k)})$, which constitute our training database. Of course, we can measure only X and deduce Y using (5.1).

We will consider a classic Neural Network architecture of the Feed Forward type, and for our tests here, with only one hidden layer. Each of our layers is fully connected to the next one (or to outside for the third layer), that is, both the neurons in the hidden and the output layers receive input from each of the neurons in the previous layer. The dimension of the input layer is equal to the number of paths P while L, the number of links, is the number of outcomes from the output layer. The dimension of the hidden layer is a variable parameter that we adjust for optimizing the performance. This happens, as expected, for a number of hidden units much larger than the dimension of the input.

4.3.3 Simulated traffic for the training phase

The training of the Neural Network requires a large volume of labeled data. In our problem, the features are the end-to-end measurements and the labels are the link metrics. Hence, constructing the dataset requires collecting exhaustive link level metrics in a small time window for each input example. This process should be repeated in different network conditions to avoid the over-fitting of the model. Thus, collecting real data for the training introduces an important overhead in the process. This scenario seems to be unrealistic since the objective of the monitoring operation is to afford useful information about the network state without disturbing the network functioning. To avoid this issue, we train our Machine Learning tool using simulated data.

The Neural Network has to learn how to approximate the link metrics from the path measurements. In other words, it must capture the spatial correlations created by the topology that enable to find relations between the end-to-end measurements and the link metrics. In addition, the Neural Network should not be over-fitted to some specific values and should take into consideration the temporal variability of the link metrics. We propose to generate exhaustive random samples of link metrics denoted X'. In fact, with each link metric, we associate a random value between 0 and B to construct one example of X'. Its associated simulated end-to-end measurement Y' is computed using $Y' = AX'$. This procedure is repeated multiple

times to construct an important number of couples (X', Y'). The computed end-to-end metrics Y' will be the features of the training, while the labels will be the generated link metrics examples X'.

4.3.4 Training step

The inference of link metrics from path measurements can be considered as a regression problem. The input is the path-level measurements and the expected output is the link-level metrics estimation. We train the Neural Network using the iterations of forward and backward propagation with the artificially created data. In our experiments, we use the classic Adam optimizer algorithm [103] for backward propagation. It is possible to pass all the training examples multiple times through the forward and backward process. An *epoch* in the learning process represents one forward and one backward propagation.

4.3.5 Testing step

After training the neural network with artificial data, we use it to estimate the link metrics from the path measurements. Observe that if there is a change in the network topology or the used paths, we have to repeat the training step taking into consideration the new updates. Observe also that this is a one-shot step, done only once. Using the trained Neural Network in the operational phase is then a quick procedure.

4.4 Model evaluation and result analysis

4.4.1 Singular Value Decomposition-based reference method

To illustrate the performances reached of our proposal, we compare it with a basic reference solution [104] based on Singular Value Decomposition (SVD) [105]: basically, every matrix A of dimension $m \times n$ can be written as $A = USU'^{\mathrm{T}}$ where U is an $m \times m$ unitary matrix (that is, $U\,U^{\mathrm{T}} = I$), S is an $m \times n$ diagonal matrix whose diagonal is composed of the singular values of A (the square roots of the eigenvalues of matrix $A^{\mathrm{T}}A$) and U' is an $n \times n$ unitary matrix. The pseudo-inverse of matrix A, denoted A^{+}, is $A^{+} = U'S^{+}U^{\mathrm{T}}$, where S^{+}, the pseudo-inverse of matrix S, is the transpose of the matrix obtained by replacing the non zero singular values of A in S by their corresponding inverses. The link metrics can then be computed by multiplying the pseudo-inverse A^{+} by the end-to-end measurement vector Y:

$$X = A^{+}Y. \tag{4.2}$$

We use this method to compare it with our proposal to show the efficiency of Machine Learning solutions to solve a classic statistical problem. Comparing the performances of the different Machine Learning models in this inference task is in the scope of our future work.

In order to evaluate the performances of our procedure TOM, we tested it using two different topologies taken from [99]. For each topology, we selected two nodes to exchange the monitoring traffic and we considered a predefined list of paths. The used paths represent the mapping between the virtual links and their corresponding representations in the shared infrastructure. The two topologies are provided with a dataset of multiple samples of delay measurements performed on the different links. The traffic was simulated with the `Omnet++4` network emulator. For each topology, we computed the end-to-end delay on each path, that is, the sum of the delays on the links composing them, and we estimated the link metrics.

The error is evaluated using (4.3):

$$\text{Error (in \%)} = 100\frac{|V^{\text{estimated}} - V^{\text{real}}|}{B}, \tag{4.3}$$

where V^{real} is the exact value of the metric V we are interested in, $V^{\text{estimated}}$ is the estimation, and B is an upper-bound of V.

The first topology, shown in Fig. 3.6, has five nodes and eight edges. The second, depicted in Fig. 3.7, is composed of nine nodes and twenty-two edges.

We made multiple tests to compare the results of our approach with the method described in Subsection 4.4.1. We also evaluate the accuracy of the estimations and the computing time as a function of different criteria like the number of layers and their sizes, the size of the training data-set, the activation function chosen and the number of epochs in the learning process. After these experiments, we concluded, for instance, that a single hidden layer was enough to obtain a good performance (see V.C.2 below).

We evaluate the performances of our solution regarding different parameters. The variation of the activation function and the number of epochs does not have a significant impact on the results. In the next tests, we use the Rectifier Linear Unit (*ReLU*) activation function and three epochs for the training phase. The errors indicated in the different figures are given in relative terms and in % as described by (4.3).

4.4.1.1 Used paths

In these illustrations, we use only one hidden layer, as previously stated, and we fix the number of examples in the training data set to $5 \cdot 10^5$.

For a fixed number of paths in one test, we select the first ones from the predefined list. We evaluate the performances of our proposal with two topologies and we compare it with the SVD-based solution. Fig. 4.4 and Fig. 4.5 illustrate the dispersion of the percentage error of the

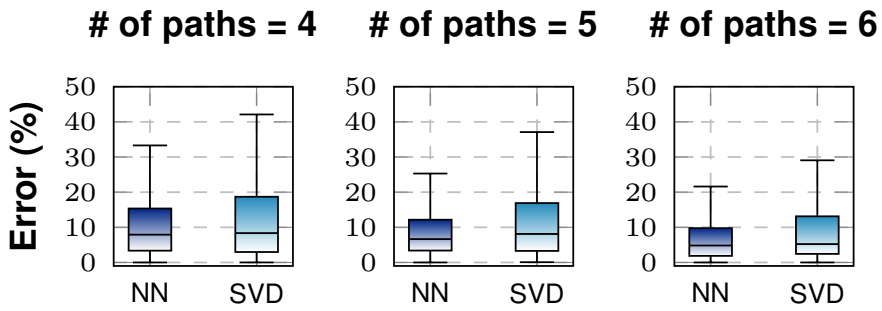


Figure 4.4 – Topology A: error vs # of paths

two solutions with the two studied topologies A and B.

Increasing the number of paths obviously enhances the accuracy of the solutions. The NN technique has always a better estimation accuracy. The difference is more visible with topology B. In fact, with topology A and 6 paths, the median of the absolute error with our Neural Network is 4.8%, while it is 5.2% with the SVD-based solution. With topology B, when we use for example 19 paths, the median is 5.9% for the NN method, while it is 22.7% when using the SVD.

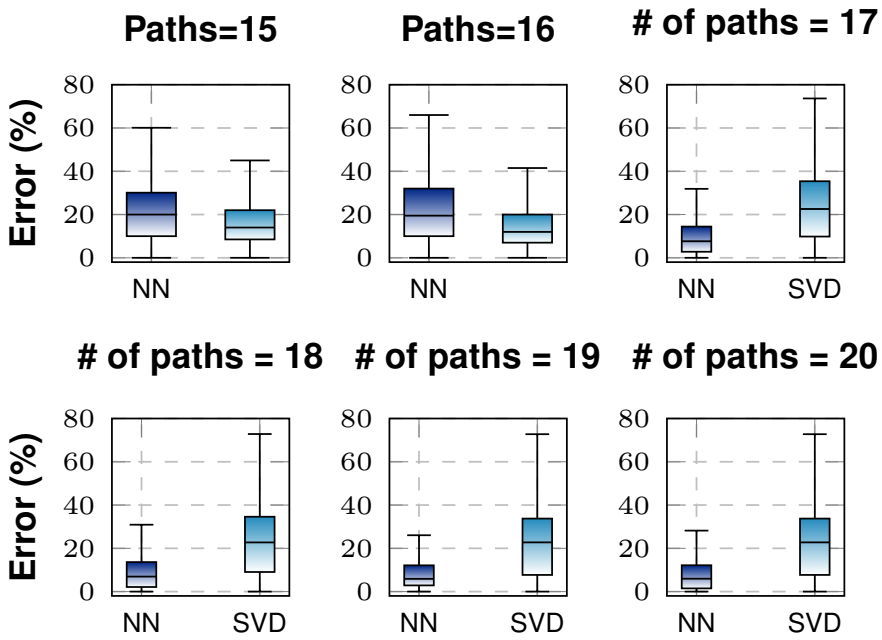


Figure 4.5 – Topology B: error vs # of paths

Using only 6 paths with topology A gives an under-determined linear equation system. Thus, multiple solutions are possible. However, there are some links that are sufficiently covered by the set of paths so that the link metric can be computed exactly. Both the Neural Network and

the SVD solutions compute these metrics with good accuracy. The difference is significant on the other links. In fact, the SVD-based approach gives one possible solution that satisfies the linear equation system, but not necessarily one close to the optimal one. The Neural Network is trained to choose the best solution that minimizes the absolute error. The difference between the two solutions is more evident when using topology B, because of the larger dimensions. The implicit covering of the correlations between the observed path metrics, given the topology of the network and the set of paths selected, done by Machine Learning technology manifests more clearly in these situations, when networks approach more realistic sizes.

4.4.1.2 Hidden layers

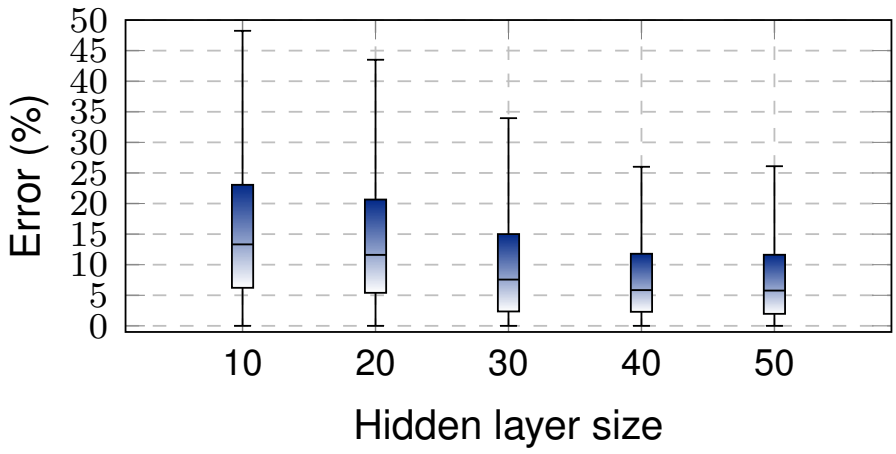


Figure 4.6 – Topology B: error vs hidden layer size

In this section, we study the impact of the number of hidden layers on the estimation accuracy and the computing time.

In these tests, we use topology B with 19 paths and the number of examples in the training data set is fixed to $5 \cdot 10^5$. The main conclusions of this set of experiments are as follows. Firstly, varying the number of the hidden layer does not impact significantly the accuracy of the estimations. That is why we used only one large middle layer in our final evaluations. This doesn't preclude future tests with deeper architectures (see the Conclusions), but our goal here is to illustrate the approach.

Secondly, we study the effect of the hidden layer size. Fig. 4.6 shows the variation of the absolute error regarding the hidden layer size. Increasing the hidden layer size enhances accuracy until reaching about 40 neurons. From this value, the error stagnates. The training time increases more or less linearly with the hidden layer size (see Fig. 4.7), increasing slightly at the end of the considered range. All these observations fit with the usual behavior of these

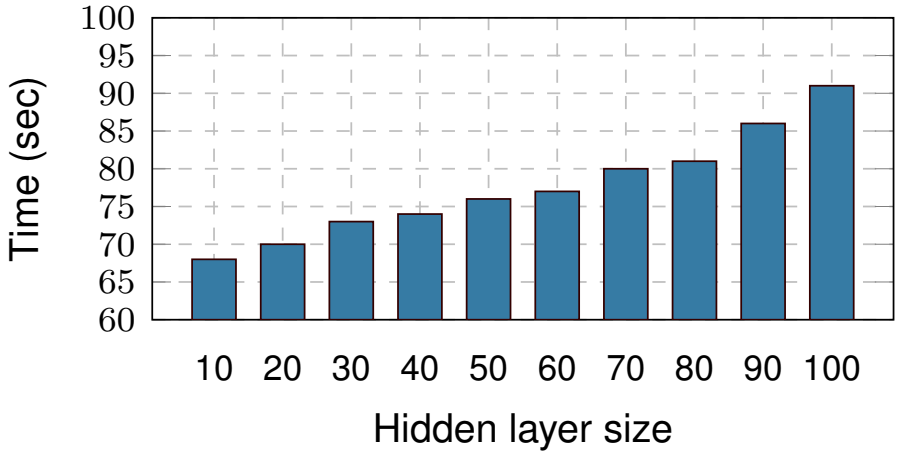


Figure 4.7 – Topology B: training time vs hidden layer size

technologies.

4.4.1.3 Training data-set

The size of the generated data-set for the training is another important parameter for the accuracy and the training time. In these tests, we use topology B with 19 paths and only one hidden layer. We study the effect of the number of training examples with three datasets of sizes 10^4, 10^5 and 10^6 respectively as described in Fig. 4.8.

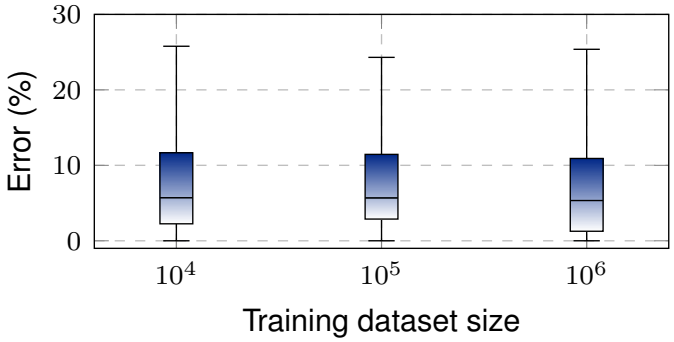


Figure 4.8 – Topology B: error vs dataset size

The training data size has a similar impact as the hidden layer size. Increasing the number of training examples enhances the accuracy of the estimations until reaching the best performance. Then, adding additional examples for the training does not impact the accuracy as shown in Fig. 4.8, it increases only the training time (see Fig. 4.9).

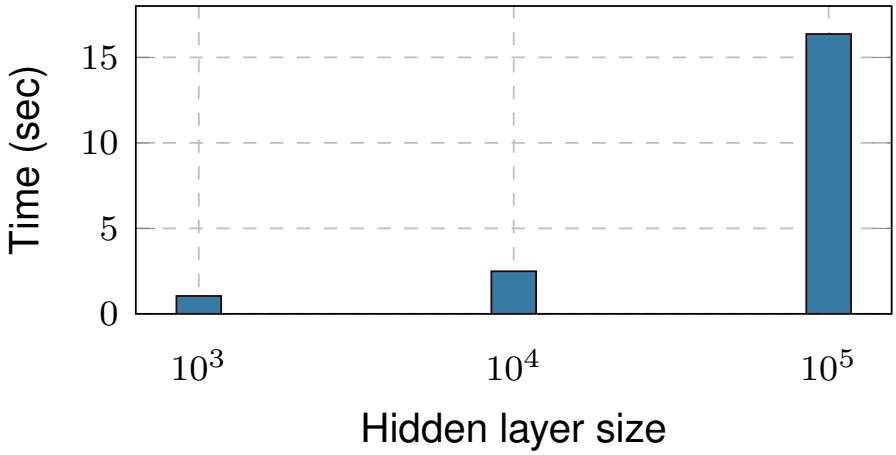


Figure 4.9 – Topology B: training time vs dataset size

4.5 A comparison between neural network and statistical methods

There are multiple criteria for comparison between the proposed approaches for link metrics inference. For example, our statistical approach gives an estimation of the probabilistic distribution of each link metric. The machine learning method we developed only provides an approximation of the link metric average. Thus, the information given by the statistical method is richer. Another criterion is the difficulty of setting up the solution. In realistic conditions, we can have additional prior information to take into consideration about the link metrics (bounds, temporal and spatial dependencies, potential failed links, or nodes, etc). Modeling all this information using classic methods is not obvious. However, it is very easy to simulate link metrics taking into consideration all the prior information that we want, and the Machine Learning approach can learn, in principle, “hidden” relations between link and paths metrics.

Fig. 4.10 illustrates the results for the same tests made with the different solutions, namely: the classic one based on SVD (Singular Value Decomposition), *k-paths*, ESA, and NN. For the tests made with the k-paths and ESA, we compute the expectation of link metrics from the computed probability distributions. The k-paths technique is tested only with granularity equal to 4 and ESA with two values, 4 and 50. We made the tests with topology B and we fixed the number of paths to 20.

The error with SDV and k-paths is very high since these solutions are not suitable for relatively large topologies. The ESA-50 and NNT solutions give usually good results and the obtained accuracy of each one are very close. However, the Machine Learning method is faster since we need the training step only once, and then, the inference step is very fast.

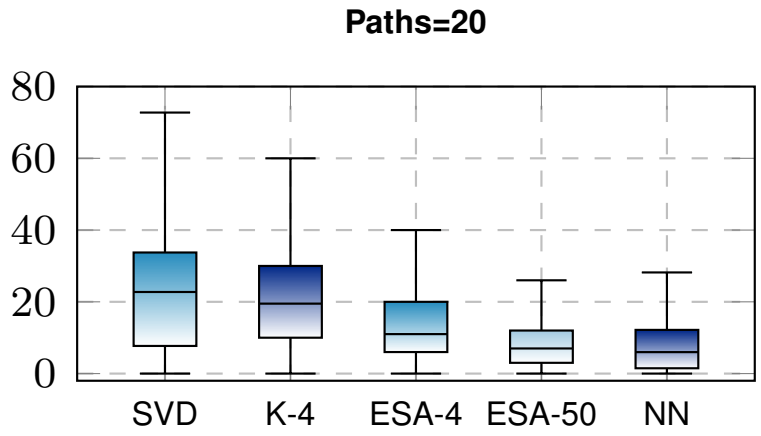


Figure 4.10 – Topology B: comparison between neural network and statistical methods

4.6 Conclusions

In this chapter, we presented a Neural Network solution for the inference of network metrics. We used additive metrics such as delays or logarithms of loss rates, leading to a linear algebraic context, but the approach can be used for other metrics such as bandwidth. One of the main features of our proposal is the use of simulated data in the training step. In addition, the learning phase can be carried out in a very short time period, which allows to easily manage changes in topologies. We used an emulated network traffic to evaluate the performances of our procedure. The results show that our Machine Learning approach gives better estimations than the pseudo-inverse statistical method.

The use of simple unicast probing paths highlights other interesting issues to study, related to the collection points placement and probing schemes configuration in the context of statistical approaches that will be studied in the next sections.

CHAPTER 5

CYCLE PROBING TOMOGRAPHY

5.1 Introduction

The probing techniques needed for tomography solutions, where the monitoring traffic is exchanged between specific nodes, i.e. monitors, are not well adapted to large backbone networks. For example, to check the operation of a specific router or link, it is very difficult to force the monitoring traffic to follow a specific path or to pass through the required network nodes due to mechanisms like multi-path routing, aggregated links and backup links [106].

Recently, more attention has been accorded to apply source routing [107] in network monitoring [108] [109]. The main idea is that the traffic source encodes the forwarding path, or some parts of the path, in the packet header. Segment Routing (SR) [110] is an efficient and flexible method of applying source routing in MPLS networks [111]. In SR, the path is encoded in the packet header as a list of segment ID (SID). Then, the packet is forwarded according to the instructions corresponding to the stacked SIDs. In spite of the flexibility offered by source routing in monitoring, the technology presents multiple challenging issues. For example, most of the commercial switches support only a limited number of SIDs in the packet headers. This adds a constraint on the length of the probed cycles. Source routing has been well explored from a traffic engineering point of view [112]. It provides fast and efficient tools to deploy traffic forwarding strategies along arbitrary paths without any additional protocol or signaling procedure. For network monitoring, this technique allows the deployment of customized probing paths. Thus, we focus in this section on the use of probing cycles.

We propose a monitoring framework based on source routing. It is composed of two main components, the “Cycle Probing” and the “Anomaly Detection” modules. For the first one, we consider minimizing the number of monitors as the main objective for the monitors’ placement strategy. Meanwhile, we compare two conditions to be satisfied: covering the network topology, and adding more conditions on the probed cycles to enhance the anomaly localization, following an algebraic modeling of the problem. The “Anomaly Detection” module is based essentially on the ESA statistical approach for additive metrics inference described in chapter 3.

The remainder of the chapter is organized as follows. Section 5.2 overviews the related work on network monitoring based on end-to-end measurements, and more specifically on the use of probing cycles. Sections 5.3 and 5.4 present respectively the problem formulation and

the framework components. Section 5.5 describes the implemented proof of concept and the performance results. Finally, Section 5.6 concludes the chapter and outlines some perspectives for this proposal.

5.2 Background and related works

In [108], the authors propose a Segment Routing monitoring solution. A single monitor is deployed in a graph centroid and the probes are forwarded through the probing cycles starting and ending at that point. The paper proposes a set of algorithms to minimize the probing cycles covering the topology and to encode them efficiently. Using only one monitor reduces the monitoring operation cost and discards the synchronization problems. However, this requires very long cycles to cover the network topology which can introduce additional bias in the measurements. Authors in [113] propose a method similar as in [108] since they use only one monitor for the probing. They focus on minimizing the cycle cover length, which is the total length of all the probed cycles, to reduce the consumed bandwidth. An Integer Linear Programming (ILP) formulation of the same problem is proposed in [109]. Other works studied the monitors' placement problem while enabling the deployment of regular paths between different nodes. Similarly, the same optimization methods are still applicable. A well detailed classification of these different approaches has been presented in [114] and [115].

In [116], the authors give necessary and sufficient conditions to identify an additive metric on all the network links for a given topology. The proposed solution identifies the minimum number of monitors to exchange probes with regular paths or cycles. However, no constraint on the length of the paths is considered. In [117], the authors propose a solution to find the optimal monitors placement for the inference of additive metrics from end-to-end measurements made on regular paths between different nodes. They give an algebraic formulation of the problem. The proposed algorithms are based on the graph decomposition into strongly connected components and find an efficient placement to guarantee the identifiability of all the links. Paper [118] studied the same problem. Meanwhile, only a subset of links is privileged to guarantee their identifiability. This leads to the objective of finding the minimal number of monitors and their right placement in order to identify an additive metric on all the considered links. In [119], the authors focus on the dual problem where the number of monitors is fixed first, and the objective is to find their effective placement to identify the highest number of links.

5.3 General context and problem description

5.3.1 General context

A network monitoring operation usually has two main steps. First, the monitoring data is collected from the network via specific tools and protocols. Second, the collected information is examined and interpreted to detect and localize the existence of malfunctioning, in order to take the needed remediation actions. Thus, our solution is composed of two main components. The first one, the “Cycle Probing”, is responsible for deploying the probing strategy and collecting the monitoring data. The second one, the “Anomaly Detection” module, is mainly based on a statistical algorithm that enables the localization of the most likely failure points from the collected end-to-end monitoring measures. As shown in Fig. 5.1, the monitoring solution can be deployed, for instance, as an SDN application over a network controller since it needs a global view of the network to deploy the probing strategy and adapt it according to the network updates.

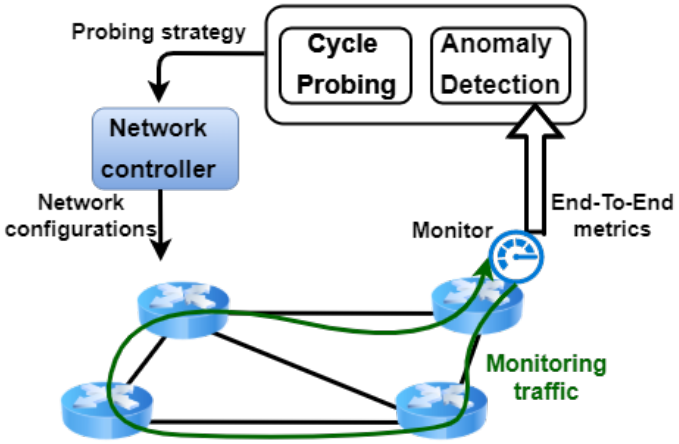


Figure 5.1 – Framework components

5.3.2 Network model and notation

The network topology is modeled by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, with set of vertices $\mathcal{V} = \{1, 2, \ldots, V\}$ and set of links $\mathcal{L} = \{1, 2, \ldots, L\}$. Let us denote by $\mathcal{C} = \{1, 2, \ldots, C\}$ the set of given probed cycles. A is a $C \times L$ Boolean matrix whose rows code the probed cycles vectors: $A(c, \ell)$ is equal to 1 if cycle c includes link ℓ. The monitors send and receive the probes through these cycles and measure the end-to-end performances. Y denotes the metrics vector of the selected probed cycles. Hence, its size is C and the cth element $Y(c)$ represents the metric on cycle c. X denotes the vector of the unknown link metrics. Thus, its size is L and its ℓth element $X(\ell)$

is the metric value corresponding to link ℓ.

5.3.3 Introducing two placement strategies

Consider the case of additive metrics, such as the delay. Using the notation described above and in chapter 2, the process of producing the end-to-end measurement linear operation:

$$Y = AX. \tag{5.1}$$

Assume that the objective of collecting end-to-end performances is to supervise the general network state and localize the failure points. Problems are simply defined by the fact that the considered metric value on a path/link (think, for instance, of delays or losses) are higher than some beforehand specified threshold. See that the metric whose value is 1 on path/link z if there is an anomaly on z, is not any more additive.

Covering all the network's links by the probing cycles allows the detection of any anomaly. From this observation, the first strategy for the monitors' placement, called **LinkCovering**, has simply the goal of covering the maximum number of links without any additional constraint. However, if a breakdown is detected, it is difficult to localize exactly the failed link. This requires computing the link metrics vector X. Now, to solve (5.1) for X, matrix A should be square and non singular. This adds new constraints to the probing cycles: we need L different cycles for which their vector representations are linearly independent. This constraint can be hard or impossible to satisfy in realistic conditions. A possible strategy is then to search the maximum of linearly independent equations to compute a good approximation of X. This leads to our second strategy for the monitors' placement, called **MaxRank**, whose goal is to choose the cycles such as the rank of A is maximized.

5.4 Cycle probing framework

Before searching the best monitors placement, the "Cycle Probing" module starts by exploring the possible cycles that can be generated from each feasible node v. Thus, we propose a resolution algorithm that explores the graph network to find all the possible cycles formed by less than k links, for a given k. In this way, the monitors' placement task can be seen as a set covering problem. For the LinkCovering strategy, the objective is to cover the set of links $\mathcal{L}$. For the MaxRank strategy, the objective is to cover the network topology with cycles that give the maximum possible rank of matrix A to enhance the anomaly localization.

There are multiple approaches to solve the set covering problem [120]. In this paper, we explore two methods. The first seeks an optimal solution using the branch and bound technique.

However, its computational complexity is very high and it is intractable with large topologies. The second is based on a greedy approach, which is much faster but not optimal.

5.4.1 Additional notation

Let $\mathcal{C}_k$ denote the set of probing cycles formed by less than k links. The subset that includes vertex v is denoted by $\mathcal{C}_k^v$. Let us denote by $\mathcal{B}_k^v$ a basis of linearly independent vectors in $\mathcal{C}_k^v$, with dimension dim$(\mathcal{B}_k^v)$. For each vertex v, the set of links covered by $\mathcal{C}_k^v$ is denoted by R_k^v. The selected nodes of $\mathcal{G}$ for the monitors' placement are denoted by $\mathcal{M}$.

5.4.2 Probing cycles encoding

Each probe stacks the forwarding path in the packet header as a list of SIDs. Most network equipment have a limitation on the number of SIDs which limits the probing cycle length. The process of describing a path as a list of segments is called path encoding.

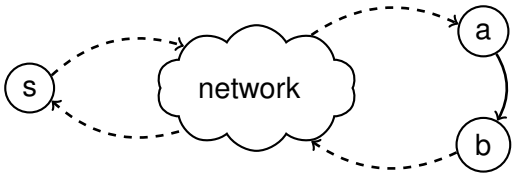


Figure 5.2 – To reach a link between nodes a and b starting from s, it is possible to forward the probe to node a with a node SID, then to b with an adjacency SID and finally to s with a node SID.

There are two types of segments: adjacency segments, when the router forward the probe to an intermediate link before achieving the final destination, or node segments, when the router sends the probe to an intermediate node.

It is possible to reach any point in the network with long paths encoded with node SIDs as proposed in [108]. However, this can affect the measurement accuracy by introducing multiple sources of bias. If we consider the example illustrated in Fig. 5.2, to forward the probes from s to a or from b to s, the routers use the shortest paths as the regular traffic. However, it is difficult to know exactly the taken paths due to the dynamic mechanisms deployed for traffic load balancing. In our proposal, the cycles are encoded with only adjacency SIDs. The complete path is described in the packet header, so all the crossed links and nodes are well known beforehand. It is possible to make a trade-off by combining node and adjacency SIDs to extend the coverage of each monitor. However, the use of node SIDs should be well controlled to avoid uncertainty and bias in the collected end-to-end measurements.

In Algorithm 4 we describe a simple recursive method to generate the possible probing cycles starting and ending at a given vertex v.

Algorithm 4 Cycles construction

INPUT: Topology $\mathcal{G}$, length k, vertex v.
OUTPUT: $\mathcal{C}_k^v$.

```
1: startNode ← v
2: C_k^v ← ∅
3: function FINDCYCLES(vertex, visited)
4:     if |visited| > k then
5:         Return
6:     end if
7:     if vertex = startNode then
8:         C_k^vertex.add(visited)
9:         Return
10:    end if
11:    for v ∈ vertex.adjacentNodes do
12:        if v ∉ visted then
13:            FINDCYCLES(v, visited ∪ {v})
14:        end if
15:    end for
16: end function
```

The algorithm starts by exploring the network graph from the node v to find all the possible cycles that start and end from this node. At each recursive call, it keeps track of the already visited nodes to avoid them later. If the starting node is reached again with less than k links, the taken path corresponds to a probing cycle.

5.4.3 LinkCovering algorithms

5.4.3.1 Optimal algorithm

We consider the monitors' placement task as a set covering problem. Thus, the objective is to minimize the number of selected monitors in $\mathcal{M}$ to cover $\mathcal{L}$ using the branch and bound technique. Finding the optimal solution for the set covering problem is NP-complete [120]. The branch and bound technique is an efficient method to deal with this complexity level by avoiding to explore some branches of the solution space that cannot lead to a potential solution.

Before exploring a branch of a solution space, the algorithm approximates the largest set of links that can be covered. The first step of the algorithm is the computation of the associated bounds of the methodology. If we consider the ordered set of vertices $\mathcal{V} = \{1, 2, ..., V\}$,

Algorithm 5 Maximum Links Covering algorithm (MLC)

INPUT: Topology $\mathcal{G}$, $\mathcal{R}_k$.
OUTPUT: Monitors placement $\mathcal{M}$.

```
1: boundLinks^{V+1} = ∅
2: for v ∈ V do
3:     boundLinks^{V-v+1} ← boundLinks^{V-v+2} ∪ R_k^{V-v+1}
4: end for
5: M ← ∅
6: maxLinks ← |boundLinks^1|
7: function MLC(coveredLinks, usedNodes, vertex)
8:     if vertex = V then
9:         if |coveredLinks| = maxLinks then:
10:            if |usedNodes| < |M| then:
11:                M ← usedNodes                          ▷ Save Solution
12:            end if
13:        end if
14:        Return
15:    end if
16:    newBound ← coveredLinks ∪ boundLinks^{vertex}
17:    if |newBound| = maxLinks then
18:        MLC(coveredLinks ∪ R_k^{vertex}, usedNodes ∪ {vertex}, vertex + 1)
19:        MLC(coveredLinks, usedNodes, vertex + 1)
20:    end if
21: end function
```

$boundLinks^v$ returns the largest set of links that can be covered by any subset of vertices included in $\{v, v+1, ..., V\}$.

The MLC algorithm (Maximum Links Covering) is implemented with a recursive function as described in Algorithm 5. At each node v, the algorithm tests if all the non covered links are included in $boundLinks^v$. If it is not the case, it means that there is no possible solution that can be founded by exploring the remainder part of the brunch. Then, there are two possible recursive calls: either node v is selected or not. The first corresponds to the case where the node v is selected. Thus, the new covered links by the monitor v are added to the covered links set. The second corresponds to the case where the node v is not selected and the recursive call takes only the set of covered links of the previous step. When the algorithm reaches the final step and all the links are covered, the set of selected nodes corresponds to a possible solution. Then, it is compared to the last saved solution to choose the best between them.

5.4.3.2 Greedy algorithm

The branch and bound method can reach a high accuracy, but its computational complexity is still very high, becoming useless in case of large topologies. Thus, we propose an alternative greedy approach described by Algorithm 6. The GMLC algorithm (Greedy Maximum Link Covering) makes multiple iterations. At each one, the network nodes are browsed to select the probing node that covers the maximum of new links. This process is repeated until the maximum possible number of covered links is reached.

5.4.3.3 Complexity analysis

The worst case complexity of the optimal algorithm is $O(2^V L)$, when all the solution space branches are explored. There are 2^V possible combinations of monitors' placements. The complexity verification of the optimality of each one is $O(L)$. The worst case complexity of the greedy algorithm is $O(V^2 L)$. At each iteration, all the nodes are browsed to find the one that covers the maximum number of links which can be done in $O(VL)$ steps. These iterations are repeated until covering all the links.

Algorithm 6 Greedy Maximum Link Covering (GMLC)

INPUT: Topology $\mathcal{G}$, $\mathcal{R}_k$.
OUTPUT: Monitors placement $\mathcal{M}$.

1: $maxLinks \leftarrow |BoundLinks^1|$
2: $\mathcal{M} \leftarrow \emptyset$
3: $links \leftarrow \emptyset$
4: **while** $|links| < maxLinks$ **do**
5: $\quad s \leftarrow s \in \mathcal{V} | \forall v \in \mathcal{V}, |\mathcal{R}_k^v \setminus links| \leq |\mathcal{R}_k^s \setminus links| \ \ \& \ \ s \notin \mathcal{M}$
6: $\quad links \leftarrow links \cup \mathcal{R}_k^s$
7: $\quad \mathcal{M}.add(s)$
8: **end while**

5.4.4 MaxRank algorithms

In this section, we focus on the second strategy for monitors placement whose goal is to maximize the rank of matrix A. It is obvious that by maximizing the rank of A, the number of covered links will be maximized.

If we consider the global set of cycles $\mathcal{C}_k$, their associated vectors can be represented in a basis denoted by $\mathcal{B}_k$ with dimension D. Therefore, the objective is to find D probing cycles, whose representative vectors are linearly independent to form the basis $\mathcal{B}_k$. Using the notation

described before, the objective is to cover the basis $\mathcal{B}_k$ with the minimal set of basis $\mathcal{B}_k^v$ which maximizes the rank of the obtained matrix A.

5.4.4.1 Optimal algorithm

The Maximum Rank algorithm (MR) applies the same approach as the optimal algorithm for LinkCovering. Meanwhile, the condition becomes maximizing the rank of the matrix A.

Algorithm 7 Maximum Rank algorithm (MR)

INPUT: Topology $\mathcal{G}$, Probing cycles $\mathcal{C}_k$.
OUTPUT: Monitors placement $\mathcal{M}$.

$boundDim^{V+1} = \emptyset$
for $v \in \mathcal{V}$ **do**
    $boundDim^{V-v+1} \leftarrow bounDim^{V-v+2} \cup \mathcal{B}_k^{V-v+1}$
end for
$\mathcal{M} \leftarrow \emptyset$
$D \leftarrow dim(boundDim^1)$
function MR($basis, usedNodes, vertex$)
    if $vertex = V$ **then**
        if $dim(basis) = D$ **then**:
            if $|usedNodes| < |\mathcal{M}|$ **then**:
                $\mathcal{M} \leftarrow usedNodes$ ▷ Save Solution
            end if
        end if
        Return
    end if
    $newBound \leftarrow dim(basis \cup boundDim^{vertex})$
    if $dim(newBound) = D$ **then**
        MR($basis \cup \mathcal{B}_k^{vertex}, usedNodes \cup \{vertex\}, vertex + 1$)
        MR($basis, usedNodes, vertex + 1$)
    end if
end function

The algorithm starts by computing the bounds used for the branch and bound technique. The objective is to cover the basis of linearly independent vectors $\mathcal{B}_k$. The bound for a subset of monitors is a basis of vectors representing the probing cycles of these nodes. $boundDim^v$ represents the basis of maximum dimension that can be formed with any subset of monitors included in $\{v, v+1, ..., V\}$. The remainder of the algorithm is very similar to the LinkCovering strategy as described in Algorithm 7.

5.4.4.2 Greedy algorithm

Algorithm 8 Greedy Maximum Rank (GMR)

INPUT: Topology $\mathcal{G}$, Probing cycles $\mathcal{C}_k$.
OUTPUT: Monitors placement $\mathcal{M}$.

1: $D \leftarrow dim(boundDim^1)$
2: $\mathcal{M} \leftarrow \emptyset$
3: $basis \leftarrow \emptyset$
4: **while** $dim(basis) < D$ **do**
5: $\quad s \leftarrow s \in \mathcal{V} | \forall v \in \mathcal{V}, dim(basis \cup \mathcal{B}_k^v) \leq dim(basis \cup \mathcal{B}_k^s) \ \ \& \ \ s \notin \mathcal{M}$
6: $\quad basis \leftarrow basis \cup \mathcal{B}_k^s$
7: $\quad \mathcal{M}.add(s)$
8: **end while**

The greedy approach of the MaxRank procedure uses the same concept as the LinkCovering strategy. The GMR algorithm (Greedy Maximum Rank) browses the graph nodes and chooses the best placement to maximize the rank of A. It makes similar iterations until covering the global basis $\mathcal{B}_k$. This process is described in Algorithm 8.

5.4.4.3 Complexity analysis

The worst case complexity of the optimal algorithm can be written $O(2^V L^3)$. In fact, there are 2^V of monitors' placement combinations. Then, the verification of the optimality of each one requires to compute the rank of a $L \times L$ matrix (generally with smaller dimensions) with a complexity $O(L^3)$.

The worst case complexity of the greedy algorithm is $O(V^2 L^3)$. At each iteration, all the nodes are browsed to find the best one that maximizes the rank of A with a complexity $O(VL^3)$. These iterations are repeated until maximizing the rank of A.

5.4.5 Anomaly Detection module

The "Anomaly Detection" module aggregates the end-to-end performances from the monitors to make a deeper analysis of the collected data. The ESA algorithm computes the probability distribution for each link metric. Then, an alarm can be activated if the failure probability on a link exceeds a fixed threshold, let's say if $\Pr(X_\ell > V_{\text{fail}}) > p^*$.

5.5 Evaluation

5.5.1 Proof of concept and testing environment

To evaluate the performance of our solution, we implement a test-bed over the `Mininet` emulator. We experiment with topologies taken from [100] and [101]. We use a P4 [121] software switch named “behavioral model” (bmv) [122] with a source routing implementation [123] to build the topologies. The switches are configured via the RYU controller.

The monitors are implemented as `Python` scripts that are running over hosts connected to the switches. The "Cycle Probing" module selects the hosts’ placements using the algorithms described above and deploy them on the network topology. Each host constructs the probes using the `Scapy Python` library [12]. The forwarding path is stacked is the packet header. The number of probes sent on each cycle is a configurable parameter. Each probe stacks in the header the forwarding path to follow the corresponding probing cycle and return to the host. Before sending each probe, the host saves the sending timestamp with the associated packet identifier. When the probe comes back to its starting node, the host saves the returning timestamp to compute the end-to-end delay. The monitor computes the end-to-end delays for the different probes and sends the information to the "Anomaly Detection" module. This entity aggregates the measurement from all the deployed monitors and crosses them to analyze the network state and localize the potential anomalies using the algorithm described in Section 5.4.5.

Meanwhile, for each experience, we simulate randomly an anomaly in the network and the framework tries to detect it. Since we are focusing on additive metrics, the chosen scenario is increasing the delay in a randomly selected link.

5.5.2 Results

5.5.2.1 Monitors placement evaluation

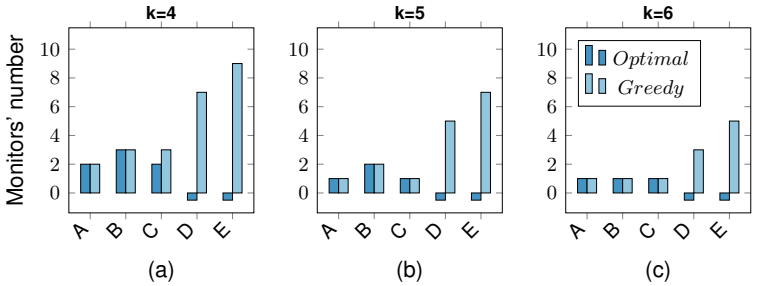


Figure 5.3 – k vs number of monitors for LinkCovering strategy.

Firstly, we compare the performances of the optimal and the greedy approaches of the

monitors' placement with different topologies with characteristics described in Table 5.1.

Table 5.1 – Network topologies

Toplogy	**A-GridNet**	**B-Polska**	**C-NewYork**	**D-India**	**E-Pioro**
Vertices	9	12	16	35	40
Edges	20	20	51	80	89

For each studied topology, we compute the minimal number of monitors to cover it, or to maximize the path matrix rank A. We vary the maximal number of labels k for path encoding.

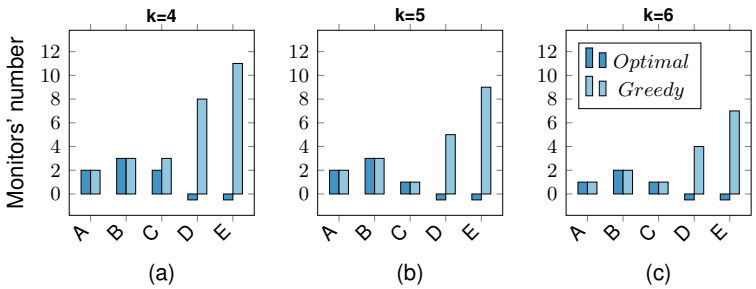


Figure 5.4 – k vs number of monitors for MaxRank strategy.

Fig. 5.3 and Fig. 5.4 illustrate the number of needed monitors for different value of k respectively for LinkCovering and MaxRank strategies.

For small and medium topologies like A, B and C, the optimal and the greedy solution usually give the same results except for a few infrequent cases like topology C with k equal to 4. With larger topologies like D and E, the optimal algorithm cannot give results in reasonable computing time.

Comparing the two strategies for the monitors' placement, maximizing the rank of A usually requires more monitors than only covering the topologies, an obvious result.

Fig. 5.5 and Fig. 5.6 illustrate the computing time for the different experiences. The computing time increases exponentially according to the number of links only with the optimal algorithm as shown with topology C compared to A and B. Remark: the computing time of the second algorithm is too small compared to that of the first; this makes often hard to represent both in the same graphics.

5.5.2.2 Anomaly detection evaluation

We choose an anomaly detection use case to test the efficiency of our monitoring solution with its two probing strategies. For each experience, we choose randomly a link to generate an anomaly and the framework tries to localize it. We use the testbed described in Section 5.5.1 to make multiple tests on the two topologies A and B. Note that we cannot test with larger

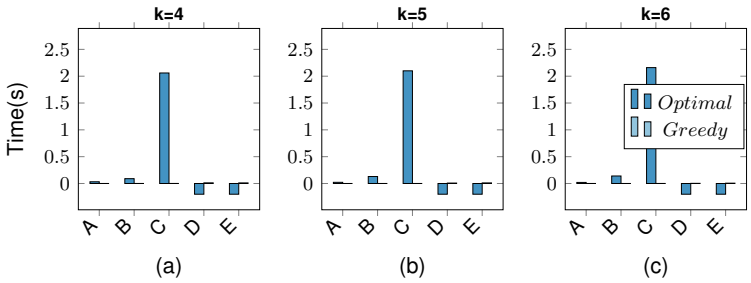


Figure 5.5 – k vs computing time for LinkCovering strategy.

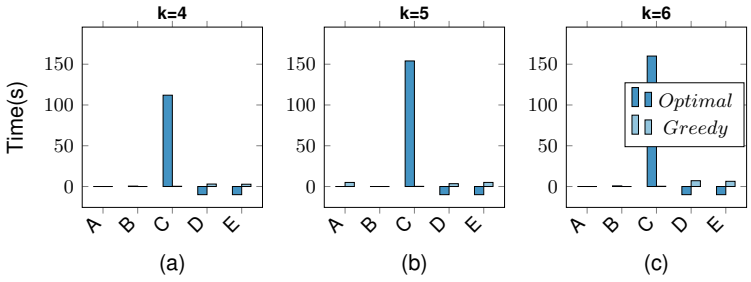


Figure 5.6 – k vs computing time for MaxRank strategy.

topologies since the used network emulator needs much more resources (CPU and memory) to emulate the needed scenarios in good conditions. However, the proposed algorithms are well adapted for larger ones.

We make 30 different experiences for each topology. In each one, the “Anomaly Detection” module computes the probability of failure on each link. Then, the alarm can be activated if this probability exceeds a fixed threshold denoted p^*. As a result, this task can be considered as a binary one, and the results for each fixed threshold can be summarized by four metrics that are described in Table 5.2.

Table 5.2 – Binary classification parameters

Variable	Description
True Positives (TP)	Successfully detected link failure
False Positives (FP)	False warning
True Negatives (TN)	Good prediction of normal links
False Negatives (FN)	Not detected link failure

From these basic parameters, multiple metrics can be computed to evaluate the accuracy of a binary classification model [124]. However, there is an important particularity in our case that must be taken into consideration. The two classes that we are considering, the normal and failed links, are heavily imbalanced. Fig. 5.7 illustrates the confusion matrix representing the four outcomes explained in Table 5.2 for 30 experiences made on topology A.

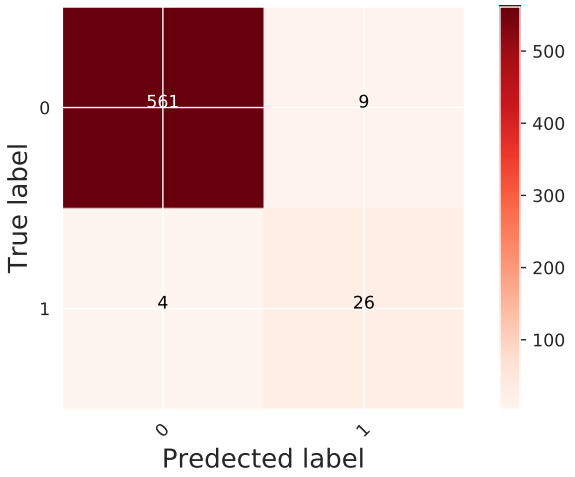


Figure 5.7 – Confusion matrix for 30 test made with topology A. The MaxRank probing strategy is used with k equal to 4. The threshold p^* is 0.5. 1 indicates the failed link class, while 0 is designated for normal links.

It is clear that the number of TN, corresponding to normal links, is largely higher than the other parameters. This unbalance is due to the process made in the experiences. Only one link is always selected to increase the delay while all the other links are not impacted. Generally, most of the negative samples (normal links) are easy to predict, especially the links that are far from the congestion zone in the network. As a result, the number of TN is always high which gives a fake indicator for a good performance of binary classification. This is a typical case usually faced while dealing with heavily imbalances classes [13]. Hence, it is not appropriate to consider the conventionally used ROC metrics (the False Positive Rate and True Positive Rate) in such a situation since they depend on the TN parameter as described in (5.2) and (5.3).

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}. \tag{5.2}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}. \tag{5.3}$$

The precision and recall metrics are well adapted to evaluate a binary classification model with imbalanced data [14]. These metrics focus more on the positive class and is not affected by the large number of TN as described in (5.4).

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN}. \tag{5.4}$$

Fig.5.8 shows the probabilities dispersion of 30 tests made with topology A. The prediction-recall curve is a model to evaluate the binary classification model. It is obtained by computing

multiple pairs of precision and recall for different thresholds p^* (represented by the dotted red line).

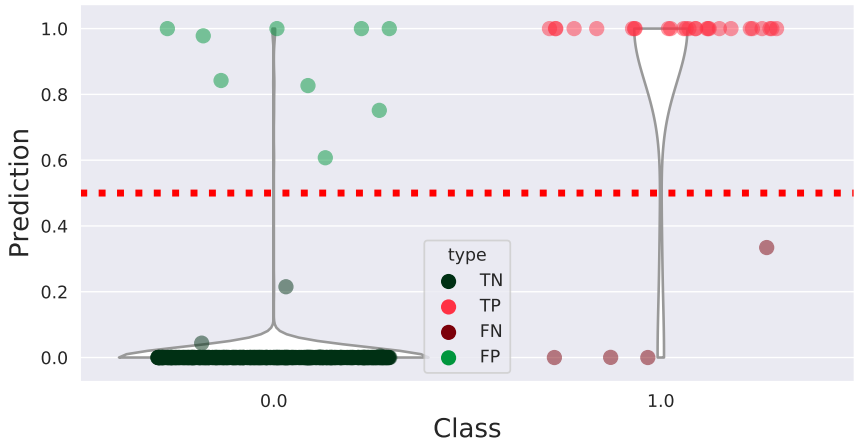


Figure 5.8 – Class distribution probabilities for 30 tests made with topology A. The MaxRank probing strategy is used with k equal to 4. The threshold p^* is 0.5.

These values are used to plot the precision-recall curve presented in Fig.5.9 for topology A. We use the AUC score (Area Under the precision-recall Curve) as a global indicator to evaluate the performances: the higher the area, the better the model accuracy. We make similar tests with the two strategies of probing. The MaxRank probing strategy outperforms the LinkCovering since its curve is always on the top. Besides, the AUC for MaxRank is 0.8, while it is 0.67 for LinkCovering.

Fig. 5.10 represents the same tests made with topology B. The MaxRank gives always better results as in the first example. For this topology, the MaxRank probing requires 3 hosts to achieve the maximum rank, while the LinkCovering requires only one host. However, with topology A, both of the two probing strategies requires two monitors, but with different placements. This case shows that even with identical resources (number of monitors), the right placement can make a significant difference.

5.6 Conclusion

In this chapter, we studied the use of source routing, particularly probing cycles, in network monitoring. We explore two different approaches for monitors placement, called LinkCovering and MaxRank, both modeled as a set covering problem, and we propose two alternatives to solve each of them: an optimal branch and bound solution and a much faster greedy algorithm. The entire solution is implemented and tested over the `Mininet` emulator to evaluate its performances.

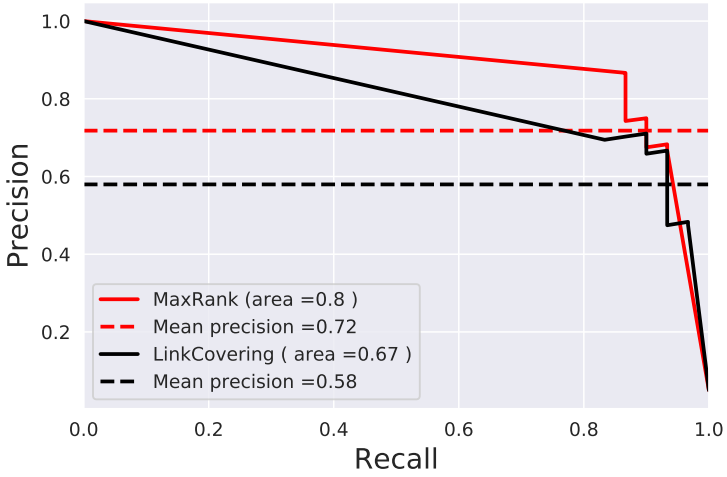


Figure 5.9 – Precision-Recall curve for 30 tests made with topology A. The two probing strategies are tested with k equal to 4.

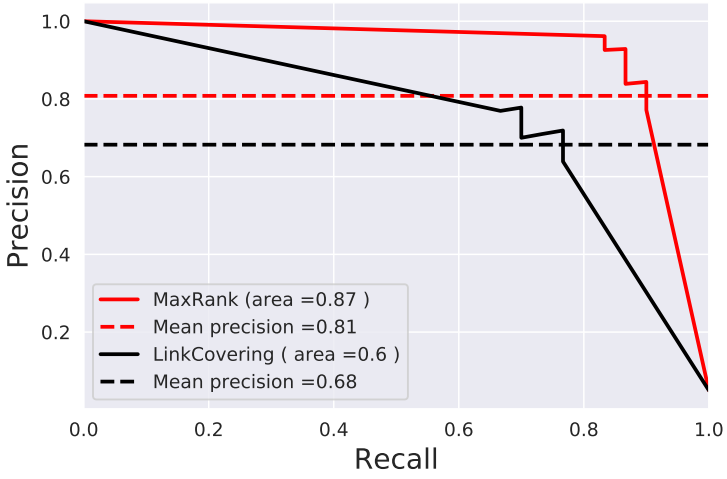


Figure 5.10 – Precision-Recall curve for 30 tests made with topology B. The two probing strategies are tested with k equal to 6.

We compare the two approaches, because most of existing solutions consist of following the same ideas as in LinkCovering. The result is that MaxRank outperforms LinkCovering and enhances the anomaly localization performances. The proof of concept with `P4` switches and a source routing implementation shows that the proposed ideas can be applied in realistic conditions. One constraint that can limit their applicability is the maximum of labels in the packet header. In this work, we consider a simple strategy for path encoding. An enhanced strategy of path encoding is possible to extend the coverage of each monitor which can reduce the cost of the probing operation and the efficiency of the anomaly detection.

The applicability to real networks of this approach relies essentially on the flexibility and the programmability offered by SDN networks. Problem sizes do not appear as a serious limitation.

CHAPTER 6

NODE FAILURE LOCALIZATION IN NFV NETWORKS

6.1 Introduction

The network slicing concept promises significant flexibility and autonomy for network management. Thanks to its main key features, the NFV and SDN technologies, the deployment of new communication services could be done faster and more flexibly. However, maintaining the reliability level of conventional networks remains a major challenge. Therefore, the monitoring of the network infrastructure dedicated to this class of services is an essential feature and becomes, at the same time, more complicated. The presence of multiple layers of abstraction in resource management and the intervention of several actors encourages the adoption of more flexible monitoring solutions like network tomography. The resources considered in virtual networks are often expressed in terms of the capacity of nodes, such as CPU and memory. In this context, it is more relevant to focus on the state of the nodes. In Boolean network tomography, multiple works studied the inference of the state of the nodes from end-to-end metrics. The main principles remain applicable to network resources abstraction in virtual networks.

This chapter describes our tomography solution, customized for the monitoring of NFV network infrastructures. First, we propose an algorithm to design the slice topology enabling to efficiently localize failed nodes. Second, we adapt our previously proposed technique for the inference of link metrics [15] to Boolean node metrics, to localize the potential failures in the network slice topology. Simulations show that our solution localizes the failed nodes with very good performance and with a small rate of false positives and false negatives.

The remainder of the chapter is organized as follows. Section 6.2 overviews the state-of-the-art on failed node localization. Section 6.3 describes the context of our contributions and the problem model formulation. Sections 6.4, 6.5 and 6.6 present respectively the main components of our solution, the slice monitoring designer and the inference algorithm. Section 6.7 provides a description of the performance evaluation method and the obtained results. Finally, Section 6.8 concludes the chapter and outlines some perspectives for future developments.

6.2 Background and related work

In this section, we study the existing strategies to identify faulty nodes in a network infrastructure.

In [125], the authors propose a Machine Learning approach for anomaly detection in virtual network functions. The proposed solution includes three main functions: anticipate failures by detecting the first signs of SLA violation, detecting the SLA violation, and identifying the root cause. This tool can help the network administrator to make efficient and fast troubleshooting and take remediation actions like rebooting a Virtual Machine (VM) or scaling the allocated resources. The approach is based on supervised learning models and the training dataset is built by monitoring data coming directly from the VMs. A fault injection process is used to emulate anomalies and provide abnormal states since the collected data obviously represents mostly the normal state. Paper [126] proposes another statistical learning solution for VNF anomaly detection. The idea here is to collect metrics by continuously monitoring the deployed VNFs and then predict the next values using a regression model. Afterward, the predicted values are compared to the reported ones in order to detect any deviation in VNFs behavior.

In [127], the proposed solution tries to find the appropriate positions of monitors and probing paths between them enabling to cover a given network topology and, then, to detect any miss behavior. The problem is formulated as an ILP model, where the goal is to reduce the monitoring cost. The problem is solved using a heuristic approach. An exact solution was proposed before in [128] for the same problem, but the methodology is intractable for large topologies.

Recently, some attention has been accorded to Boolean network tomography. The main idea is to infer the state of single points or links in a network from end-to-end Boolean metrics. In [129] the authors introduce the application of Boolean network tomography for node failure localization in a general network topology. They first define an *"identifiability"* metric to measure the maximum number of simultaneously failed nodes in a given topology. Then, they give necessary and sufficient conditions to localize failure points under different probing schemes. In [130], the authors studied a similar problem. The proposed solution gives some insights for the design of monitoring strategies that maximize the number of identifiable nodes under different probing scenarios. Finally, let us cite [131] that studied the impact of topology properties on the maximal identifiability. They establish a relation between the different typologies classes (directed, undirected, trees, d-dimensional grids, and bounded-degree graphs) and the identifiability of Boolean metrics.

6.3 General context and problem formulation

6.3.1 Network slicing

6.3.1.1 Network Function Virtualization (NFV)

One of the main features enabling network slicing is the Network Function Virtualization (NFV) concept. Classic network functions are implemented as software, called Virtual Network Functions (VNFs), that can be executed over Off-The-Shelf servers. Theses VNFs are deployed on Virtual Machines (VMs) or containers. Each network service is decomposed into a number of VNFs organized sequentially, say N of them, as a chain. Then, for each customer service request, the N VNFs are placed on N VMs. A tunnel connecting the VMs of each chain associated with the service is created. See Fig. 6.1 where we represented a service decomposed into $N = 3$ VNFs. Different requests make that a total number of 8 VMs are used to provide service to different customers. More on this in Subsection C.

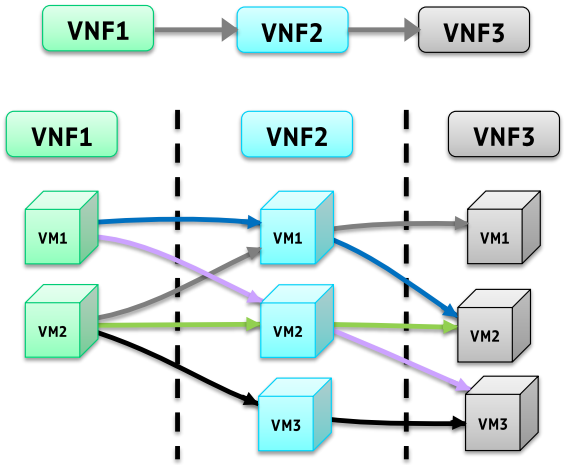


Figure 6.1 – Virtual Network Function Infrastructure example: The service is composed of three VNFs. Each one is hosted by multiple VMs. For each client, a tunnel is created between three VMs to compose the service.

6.3.1.2 Network resource sharing

Each network service is created from a network service template which provides the structure and the needed network characteristics. Thus, for a given network service, all the related instances are composed of the same VNFs. However, they can be deployed over different nodes of the NFVI.

The network slicing concept enables the design of logical networks over the shared physical infrastructure [22]. There are three layers involved in the network slicing process: the service instance layer, the network slice instance layer, and the resource layer. The network resource sharing concept is present at different levels of the network virtualization process enabling more flexibility in the network resource management.

6.3.2 Failure detection in the network resource layer

Fig. 6.1 illustrates a simple example of a shared network infrastructure between multiple instances of the same service composed of three VNFs where each VNF can have multiple instances. For each service, a tunnel is created between three VMs. The representation of the mapping of the created slices over the NFVI can look like the graph shown in Fig. 6.1.

If there are some failed nodes in the NFVI, efficient troubleshooting to identify the failure root causes can be a complex task. In such cases, the use of end-to-end metrics can be a relevant alternative to enhance the efficiency of the monitoring system. The misbehavior of a VNF can result from memory leaks, disk access problems, network problems, heavy workload, etc [125]. A failed node in the NFVI can lead to the disrupt of all the service flows passing through. Therefore, we can take advantage of the end-to-end collected metrics to cross them and infer the state of intermediate nodes. However, there are some required conditions to fulfill while designing the monitoring system in order to ensure the efficiency of such methods (see below).

The efficiency of a tomography monitoring system can be evaluated by its capacity to localize accurately the maximum of simultaneously failed points. Our objective here is to determine the necessary and sufficient conditions to detect up to k simultaneous failed points in an NFVI hosting multiple instances of one network service. Then, we shall benefit from these conditions to design monitoring strategies for NFV networks.

6.3.3 Network model and notation

Tab. 6.1 summarizes the notation adopted in this chapter. We consider a network service decomposed into a set $\mathcal{N} = \{\mathcal{N}^1, \ldots, \mathcal{N}^N\}$ of N VNFs. To respond to many service requests, the VNFs are instantiated over a set $\mathcal{V}$ of VMs. We denote by $\mathcal{V}^i$ the set of VMs where the VNF $\mathcal{N}^i$ is instantiated. The cardinalities of $\mathcal{V}$ and $\mathcal{V}^i$ are denoted by V and V^i respectively. The whole system is called the NFV Infrastructure (NFVI).

The tunnels make that there is no connection between VMs inside a same $\mathcal{V}^i$ (no "vertical" links in the picture of Fig. 6.1). Links only exist between VMs in $\mathcal{V}^i$ and $\mathcal{V}^{i+1}$ for $i = 1, 2, \ldots, N-1$. Of course, every VM in $\mathcal{V}^i$ has at least a connection with another VM in $\mathcal{V}^{i+1}$, $1 \leq i \leq N-1$ (if $N \geq 2$). We denote by $\mathcal{L}$ the set of all links (size L), and by $\mathcal{G}$ the graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$.

Table 6.1 – **List of main used variables**

variable	description
$\mathcal{N}$	set of *VNFs* composing the service; $\mathcal{N}^i$: the i^{th} VNF
$\mathcal{V}^i$	virtual network function instances representing $\mathcal{N}^i$
$\mathcal{V}^i_j$	instance j of $\mathcal{V}^i$
$\mathcal{L}_{i,j}$	set of links between $\mathcal{V}^i$ and $\mathcal{V}^j$
$\mathcal{V}, \mathcal{L}$	set of VMs and set of links between them
$\mathcal{P}$	path set, P paths
$G = (\mathcal{V}, \mathcal{L})$	network topology
$\mathcal{F}$	set of failed nodes; cardinality of $\mathcal{F} = F$

A node $\mathcal{V}^i_j$ can be either in a failed state (or down) or in a normal state (or up). We denote by X the Boolean vector representing the states of all the nodes, thus a vector with size V. The binary value 1 corresponds to the failed state and 0 to the up state. Let us denote by p a path in the graph corresponding to a chain of VMs composing the service. It is represented by a Boolean vector having size V, where component i, $p(i)$, is 1 if node i belongs to p, and to 0 otherwise. The state of a traffic path is up if all the nodes composing it are up. Otherwise, it is a failed path. We denote by Y the Boolean vector representing the paths' states. The goal is to estimate the states of the intermediate nodes from the end-to-end path metrics. Let $\mathcal{P}$ denote the set of such paths, with cardinality P, denoted $p^1, p^2, \ldots, p^P$. Let A denote the Boolean matrix whose rows are the P path vectors. Thus, $A(i, j)$ is equal to 1 if node j belongs to path p^i, and to 0 otherwise.

With this notation, between the nodes' states in X and the path states' in Y, we have the relation

$$A \odot X = Y, \tag{6.1}$$

where $\odot$ denotes the Boolean matrix product. $Y(r)$, the state on the rth path (row) is 0 if all the nodes composing it are up, i.e., $Y(r) = \vee_{j=1}^{V}(A(r, j) \wedge X(j))$. In next section, we describe how to build a topology that guarantees the localization of a maximum number of faulty nodes.

6.4 The Network Slice Designer

This section is the central one in the chapter. In the first subsection, we discuss about the tomography problem considered in the section, that will make the connection with our proposal that starts in Subsection B. Subsection C discusses some aspects of the algorithm described in previous subsection, and D discusses complexity issues.

6.4.1 Necessary and sufficient conditions for k-failures detection

Recall that our network has a particular topology as illustrated in Fig. 6.1. The end-to-end paths start from a VM on the left side (in the subset of nodes $\mathcal{V}^1$), and ends on some VM on the right side (in $\mathcal{V}^N$). Let us call here *blocks* the N sets of nodes (or VMs) $\mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^N$. Once the links built between the VMs instantiating the VNFs that compose the considered network service, we can monitor any path following any sequence of nodes $v_1, v_2, \ldots, v_N$, where v_i belongs to block $\mathcal{V}^i$, provided there is a link (belonging to some tunnel) from v_i to v_{i+1}, $i = 1, 2, \ldots, N-1$, even if $(v_1, v_2, \ldots, v_N)$ is not a tunnel itself (that is, in Fig. 6.1, even if the links don't have the same color).

In the sequel, we were inspired by the developments made in [129] where the problem of node failures detection is considered in a general setting and for an arbitrary network topology. As in other works related to the failure of network components (for instance, in fault tolerant design, as in [132]), it is often useful to decompose the problem in the cases where the number of failures is bounded by some integer k. The failure of a node or a link is always an event having a small probability, and the simultaneous failures of two or more components is much smaller, even if the considered events are not necessarily independent. So, we will organize the discussion by considering as a global assumption that the number of failures, here of node failures, is $\leq k$ for some fixed integer $k \geq 1$.

Since we are in a Boolean world, observe that when a path is measured as up (value 0), we know that all its nodes are necessarily up. Now, for the design of our approach we focus on the following property: for a selected node v, we consider the fact that for some path p through v, all nodes excepting v are up. In that case, the state of p is the state of v: if v is up, then all nodes in p are up and so is p, and when v fails, then p fails as well. The next propositions provide conditions under which this property holds.

Proposition 1 (Sufficient condition)**.** *Let us assume that the number of failed nodes is $\leq k$, for some integer $k \geq 1$. If for all node $v \in \mathcal{V}^i$ its interior degree (for $i \geq 2$) and its exterior degree (if $i \leq N-1$) are both $> k$, then there exist a path p containing v such that its state is the state of v.*

Proof. Say $v \in \mathcal{V}^i$. If $i \geq 2$, since v has at least $k+1$ neighbors in $\mathcal{V}^{i-1}$ and since there are at most k nodes down, there is at least one of these neighbors up; let us denote it by n_{i-1}. In the same way, n_{i-1} has at least one neighbor up in $\mathcal{V}^{i-2}$, and so on. The same reasoning is valid when we go to subsets $\mathcal{V}^{i+1}$, $\mathcal{V}^{i+2}$, etc. So, we can build a path $p = (n_1, n_2, \ldots, n_{i-1}, v, n_{i+1}, \ldots, n_N)$, where all nodes except v are up. This means that the state of the path coincides with the state of v: if v is up, p is up, and if the node is down, p is down as well. □

Proposition 2 (Necessary condition)**.** *Let us assume that the number of failed nodes is $\leq k$, for some integer $k \geq 1$. A necessary condition to have the property that for all node v there*

exist a path p containing v such that p's state is equal to v's state, is that for every node in $\mathcal{V}^i$ its interior degree (for $i \geq 2$) and its exterior degree (for $i \leq N-1$) are both $\geq k$.

Proof. The proof is by contradiction. We will denote by $deg^-(v)$ the in-degree or interior degree of node v, and by $deg^+(v)$ its out-degree or exterior degree.

Consider first the case of $v \in \mathcal{V}^i$ for some $i \geq 2$, with $deg^-(v) < k$. The existence of path p passing through v with all nodes up except possibly v is in contradiction with previous assumption, because it can happen that all $v' \in \mathcal{V}^{i-1}$ connected to v are down (this is because we can have up to k failed nodes and we assumed that $deg^-(v) < k$). The case of $v \in \mathcal{V}^i$ for some $i \leq N-1$ with $deg^+(v) < k$ is treated exactly in the same way. □

Previous properties are related to more general ones in [129]. In that paper, the discussion covers arbitrary topologies and, as a consequence, the development is more complex, requiring other auxiliary concepts related to the problem. Here, our specific architecture of the NFVI allows us to exhibit very simple properties that lead to the procedures that we describe in next subsection.

Let us denote by d the smallest degree in or out in graph $\mathcal{G}$: $d = \min\{deg^-(v),\ deg^+(v),$ for all node $v\}$. Observe that the sufficient condition in Proposition 1 means that $k \leq d-1$ and the necessary one in Proposition 2 means that $k \leq d$.

6.4.2 Network slice design as a matching problem

Let us consider two consecutive blocks $\mathcal{V}^i$ and $\mathcal{V}^{i+1}$ and the links built from the first to the second, which we denote by $\mathcal{L}_{i,i+1}$. Consider the problem of selecting a subset of these links in order to fulfil the sufficient condition given in Proposition 1. In other words, we consider here that we can take any existing link from any node in $\mathcal{V}^i$ to any other node in $\mathcal{V}^{i+1}$ and we want to select a subset satisfying the sufficient condition. This means that we have chosen some k satisfying $k \leq d-1$ where d is defined considering all existing links before this selection operation, and that in the given graph, all degrees are $\geq k+1$.

Previous selection operation between two sets is an instance of a classical matching problem in graph theory, typically solved using a Max-Flow technique. We will solve the problem in this way. Let us describe then the operations to be performed, which are illustrated in Fig. 6.2 using the case of $k = 1$.

The **first step** of the process is to add two "fictitious" nodes to the two sets, named Source (Sr) and Sink (Sk), as illustrated in Fig. 6.2b. The source Sr is connected to all the nodes in $\mathcal{V}^i$ with a flow capacity equal to $k+1$ (= 2 in the example), and similarly for Sk and $\mathcal{V}^{i+1}$. As typically done in matching graph problems, the capacity of the links from $\mathcal{V}^i$ to $\mathcal{V}^{i+1}$ is 1. This limits the number of links going in and out each node to be no more than $k+1$.

In the **second step**, we use a conventional Max-Flow algorithm, like the Ford-Fulkerson or the Edmonds-Karp algorithms [133] to find a maximal flow from Sr to Sk, which we denote by $\mathcal{M}^0_{i,i+1}$. In the obtained flow, if we remove from the graph the arcs where the flow function is 0, the degrees calculated with the remaining arcs do not necessarily satisfy the sufficient condition. In the example given in Fig. 6.2, where we color in red the selected arcs (see Fig. 6.2c), the In-degree of $\mathcal{V}^{i+1}_2$ and $\mathcal{V}^{i+1}_3$ is only 1, since the number of links going out of $\mathcal{V}^i_0$ and $\mathcal{V}^i_1$ is limited to 2. We need more links from and to these nodes.

To avoid such a situation, in the **third step** we add some capacity to specific arcs as illustrated in Fig. 6.2d. For each node v in $\mathcal{V}^i$ where $deg^+(v) < k+1$, we add capacity to all the arcs from the nodes in $next(v)$ to the sink Sk, where $next(v)$ denotes the nodes in $\mathcal{V}^{i+1}$ that receive arcs from v. Similarly, for each node v in $\mathcal{V}^{i+1}$ for which $deg^-(v) < k+1$, we add capacity to all edges from the source Sr to the nodes in $prev(v)$, where $prev(v)$ denotes the nodes in $\mathcal{V}^i$ that send arcs to v. We actually set the new capacity value of all these arcs to ∞ since there is no risk to select new useless arcs in $\mathcal{L}_{i,i+1}$.

Then, in the **fourth step** illustrated in Fig. 6.2e, we compute for a second time the MaxFlow between Sr and Sk. It is worth noting that the arcs selected in the first call to the MaxFlow algorithm (in the second step) cannot be reused. The obtained maximum flow is denoted by $\mathcal{M}^1_{i,i+1}$.

This matching process is repeated for each two consecutive blocks. The final obtained graph, denoted by $\mathcal{M}$, comes from composing the optimal matching obtained for each pair of consecutive blocks. We will use this new graph, a partial graph of the initial one (same nodes, part of the links) to localize up to k failures in the considered network.

6.4.3 Analysis of the selection algorithm

The matching between $\mathcal{V}^i$ and $\mathcal{V}^{i+1}$ is considered as optimal if it satisfies the two following conditions:

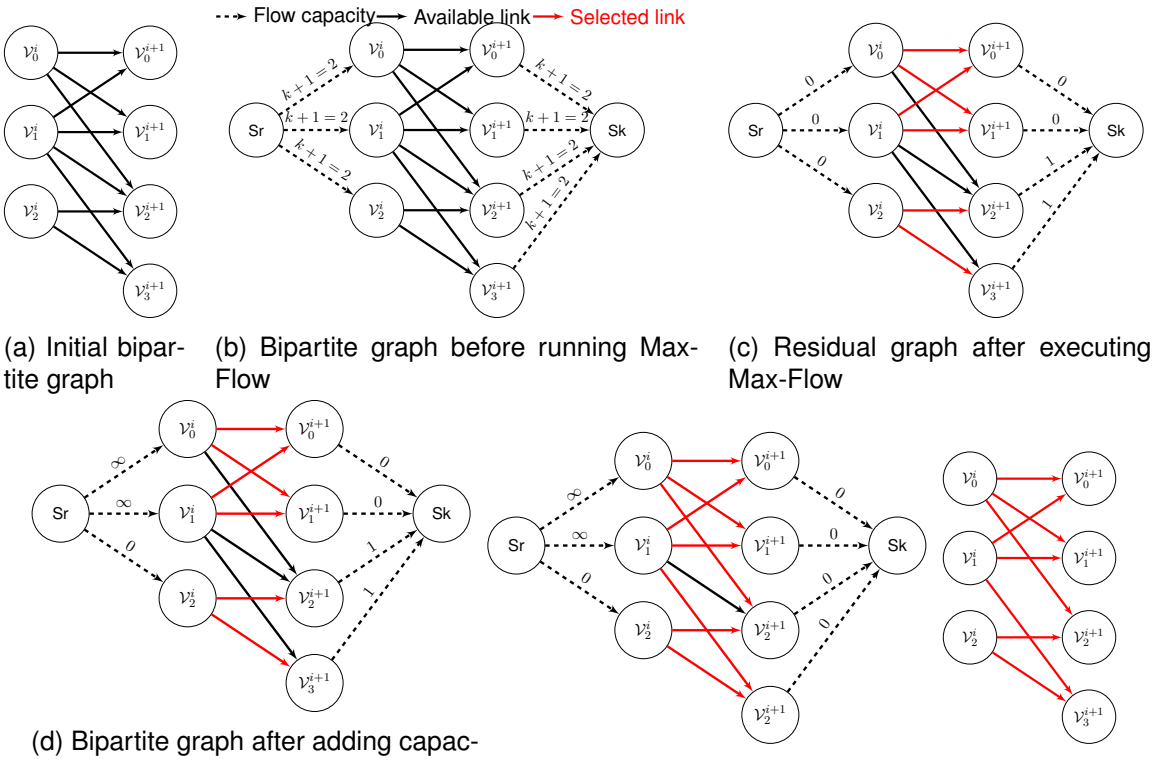
(i) $\forall v \in \mathcal{V}^i$, $deg^+(v) > k$ and $\forall v \in \mathcal{V}^{i+1}$, $deg^-(v) > k$.

(ii) There is no matching that satisfies (i) with fewer selected links.

The matching algorithm follows a greedy approach at each step with the objective of finding a globally optimal solution. The two main steps are essentially the two calls to the Max-Flow algorithm to build $\mathcal{M}^0_{i,i+1}$ and $\mathcal{M}^1_{i,i+1}$.

Let us consider the bipartite graph $\mathcal{B}^0_{i,i+1}$ composed of blocks $\mathcal{V}^i$ and $\mathcal{V}^{i+1}$ and the selected arcs after the first Max-Flow execution (corresponding to the "red part" of Fig. 6.2c). Denote by $\mathcal{B}^1_{i,i+1}$ the corresponding bipartite graph obtained after the second Max-Flow computation (Fig. 6.2f). We show first that if $\mathcal{B}^0_{i,i+1}$ is included in an optimal solution, then $\mathcal{B}^1_{i,i+1}$ will be optimal. Then, we verify that $\mathcal{B}^0_{i,i+1}$ is effectively included in an optimal solution.

Let us consider the pair of blocks $\mathcal{V}^i$ and $\mathcal{V}^{i+1}$ in $\mathcal{B}^0_{i,i+1}$. We can classify the nodes in the first block in three classes denoted $\mathcal{U}^i$, $\mathcal{W}^i$ and $\mathcal{Z}^i$ and similarly for $\mathcal{V}^{i+1}$ (the three classes are $\mathcal{U}^{i+1}$,



(a) Initial bipartite graph

(b) Bipartite graph before running Max-Flow

(c) Residual graph after executing Max-Flow

(d) Bipartite graph after adding capacity to some arcs and before calling Max-Flow a second time

(e) Residual graph after second Max-Flow

(f) Bipartite result

Figure 6.2 – Matching algorithm illustration. The values on the dotted arcs are capacities, either the initial ones, or the residual ones after a Max-Flow execution. The arcs between blocks have obviously capacity 1.

$\mathcal{W}^{i+1}$ and $\mathcal{Z}^{i+1}$) as follows: In $\mathcal{V}^i$, $\mathcal{U}^i$ is the set of nodes v such that $deg^+(v) < k+1$; $\mathcal{W}^i$ is the set of nodes v such that $deg^+(v) = k+1$ and that there are still not selected links from v; $\mathcal{Z}^i$ is the set of nodes v with $deg^+(v) = k+1$ and such that all links from v have been selected. In $\mathcal{V}^{i+1}$, $\mathcal{U}^{i+1}$ is the set of nodes v such that $deg^-(v) < k+1$; $\mathcal{W}^{i+1}$ is the set of nodes v such that $deg^-(v) = k+1$ and that there are still not selected arcs to v; $\mathcal{Z}^{i+1}$ is the set of nodes v with $deg^-(v) = k+1$ and such that all arcs arriving at v have been selected.

The objective of the next step is to select nodes from $\mathcal{W}^i$ to be connected to nodes in $\mathcal{U}^{i+1}$ and nodes in $\mathcal{W}^{i+1}$ to receive arcs from nodes in $\mathcal{U}^i$, in order to achieve the degree $k+1$ for all nodes. Thus, if $\mathcal{M}^0_{i,i+1}$ belongs to an optimal solution, $\mathcal{M}^1_{i,i+1}$ will complete it with a minimum number of links and $\mathcal{M}_{i,i+1}$ will be optimal.

Now, we show that $\mathcal{M}^0_{i,i+1}$ is included in an optimal solution. Thus, we consider an optimal matching $\mathcal{M}_{i,i+1}$. The nodes in $\mathcal{V}^i$ can be classified in three sets (the same for $\mathcal{V}^{i+1}$):

— Nodes with degree $k+1$ only linked to nodes with degree $k+1$.

— Nodes with degree $> k+1$ only linked to nodes with degree $k+1$.

— Nodes with degree $k+1$ only linked to nodes with degree $> k+1$.

If we remove the links in $\mathcal{M}_{i,i+1}$ to have a $k+1$ degree for all the nodes, these three sets will correspond to $\mathcal{U}^i$, $\mathcal{W}^i$ and $\mathcal{Z}^i$. The matching between them corresponds to a Max-Flow $\mathcal{M}^0_{i,i+1}$.

6.4.4 Complexity analysis

The computation complexity of the described process is essentially determined by the Max-Flow algorithm. If we denote by L_{moy} the average number of links between each consecutive pair of sets $\mathcal{V}^i$ and $\mathcal{V}^{i+1}$, and by V_{moy} the average number of vertices in each set $\mathcal{V}^i$, the complexity of the max-flow algorithm (Edmonds-Karp for example) can be approximated by $O(V_{moy}L^2_{moy})$. It will be invoked twice. This process is repeated $N-1$ times, corresponding two each pair of VNFs. Indeed, the global complexity of can be approximated by $O(2NV_{moy}L^2_{moy})$.

As a result, the computing time increases polynomially with the average number of edges between two VNF sets L_{moy} and linearly with the other parameters V_{moy} and N.

6.5 Boolean metrics inference

Typically, a monitoring operation consists of two main steps: the collection of measurements and information from the network, and their analysis. Thus, after defining a strategy for the design of topologies enabling the detection of failures and the deployment of the necessary probing paths, a tool is needed to exploit these end-to-end measurements. To this purpose, we propose a genetic algorithm for the inference of node states from end-to-end Boolean metrics. It is basically an adaptation of the ESA solution (Evolutionary Sampling Algorithm) proposed for the inference of *additive* link metrics in [15], but the principle remains applicable to other network metrics such as Boolean ones.

The state of each node can be either failed (1) or operational (0) and the objective of the inference algorithm is to compute the probability of each state.

As described in Section 6.3.3, X is a random variable representing the nodes states. We consider in our approach that X follows a discrete probability distribution α, seen as an $V \times 2$ matrix. Hence, $\alpha(v,0)$ represents the probability of the up state of node v and $\alpha(v,1)$ the probability of v's failure state. Our objective is then to get an accurate approximation of the probability distribution α. Observe that the elements of each row of matrix α sum up to 1.

The ESA algorithm is a population-based approach where a random population of solution vectors X is initialized and then enhanced iteratively. First, α is initialized uniformly. Then, a large number of solution vectors are generated according to the distribution α, denoted $\mathcal{X}^{rand}$. After that, only the samples that respect the inequality $||A \odot X - Y|| < \varepsilon$ are selected, where $||\cdot||$ denotes a standard matrix norm and ε an error bound that will be minimized. In the next step, a

new distribution α is computed from the selected samples, leading to an enhanced distribution. These two steps are repeated until the convergence of α. At this stage, the error bound ε is reduced and we repeat the previous steps to further improve the estimations. Meanwhile, if α doesn't converge, ε is increased to lighten the constraints. For more details about the ESA approach, see [15] and chapter 3, for our Boolean metrics case, see Algorithm 9. Observe that the situation is more favorable for the latter type of metric, since here we have only two states which reduces considerably the solution space and then the computing time.

6.6 Probing paths selection

The procedure described in Section 6.4 guarantees that the designed topology enables the accurate localization of up to k simultaneous failed nodes, i.e. there is a set of probing paths that enables a unique identification of the nodes' states from the collected end-to-end measurements. Meanwhile, we need a concrete strategy for selecting the minimal path set enabling the identification of the failed nodes. Thus, we propose in this section an adaptive approach to select the needed set of probing paths.

The first step of the monitoring operation is to cover the set of nodes $\mathcal{V}$ with a minimum number of paths from an exhaustive list denoted by $\mathcal{P}_{global}$. This enables the detection of any misbehaviour in the network topology. The process is described in lines 2-5 of Algorithm 10. It is based on an iterative process where we select at each iteration the path p that covers the maximum number of new nodes denoted by V^p until covering $\mathcal{V}$.

The performed end-to-end measurements on this initial set of paths allow the detection of any anomaly in the network. Then, we need additional paths for the localization. The process of probing path selection for faulty nodes localization is a particular case of Group Testing Theory [134]. We follow an adaptive incremental method, where we select at each iteration the path that gives the maximum of useful information according to the already measured paths.

In our path selection strategy, we privilege those who are likely in an up state. Indeed, if the end-to-end measurement on a path is 'up', we can conclude that all the nodes composing it are functional, which is not the case if the path is faulty. Thus, we compute at each iteration a score for the non selected paths to choose the best one. This score reflects the expectation of the number of 'up' nodes for which we do not know their state weighted by the probability that the path is 'up'. This estimation is computed with Formula (6.2), where $\mathcal{V}'$ denotes the nodes with unknown states. At each iteration, the value of α is computed with Algorithm 9.

$$\mathbb{E}(p) = \left(\sum_{p \in V'} \alpha(0, p) \right) \times Prob(p_{state} = 0). \tag{6.2}$$

The complete paths selection process is described in Algorithm 10.

Algorithm 9 Evolutionary Sampling Algorithm (ESA)

INPUT: Linear equation system $A \odot X = Y$
OUTPUT: α

Initialize α (α^0), ε and $\beta = 0.5$
iter = 0 ▷ iteration index
— Step 1:
while iter $\leq$ itermax **do**
    iter = iter +1
    Generate n random samples of solutions according to distribution α.
    Select the ones that respect $||A \odot X - Y|| < \varepsilon$; put them in set $\mathcal{X}^{\text{selected}}$.
    If $|\mathcal{X}^{\text{selected}}|/|\mathcal{X}^{\text{rand}}| < D$ repeat Step 1. ▷ D is a fixed threshold.
    Compute the new α (α^{iter}) from the selected samples:
        $\alpha^{\text{iter}}(\ell, j) = \frac{|x \in \mathcal{X}^{\text{selected}},\, x[\ell]=j|}{|\mathcal{X}^{\text{selected}}|}$
    if $\alpha^{iter} \approx \alpha^{iter-1}$ **then**
        Break ▷ α has converged
    end if
end while
— Step 2:
if α has converged **then**
    $\varepsilon' = \varepsilon$
    α' = α
    ε = ε β
    iter = 0
    Repeat from Step 1
else
    β = $\beta + (1 - \beta)/2$
    $\alpha = \alpha'$
    if $\beta \approx 1$ **then**
        Return α ▷ ε has converged
    else
        $\varepsilon = \varepsilon' \beta$
        iter = 0
        Repeat from Step 1
    end if
end if

6.7 Evaluation

6.7.1 Methodology

Our solution allows to design a probing strategy enabling to localize accurately a maximum number of failed nodes. To evaluate the robustness of the designed slice topologies, we proceed

Algorithm 10 Paths selection algorithm

INPUT: $\mathcal{G}$, $\mathcal{V}$, P_{global}
OUTPUT: $P_{selected}$.

$P_{selected} \leftarrow \emptyset$
while $\mathcal{V}$ is not covered **do**
    Select $p \in P_{global} | V^p \geq V^{p'}, \forall p' \in P_{global}$
    $P_{selected}.add(p)$
    $P_{global}.remove(p)$
end while
$Y \leftarrow getMeasurement(P_{selected})$
if $\exists p \in P_{selected} | Y^p = 1$ **then**
    while Failures are not localized **do**
        Initialize α
        $\mathbb{E} \leftarrow upNodesEstimates(P_{global}, \alpha)$ ▷ Computed with formula 6.2
        $p^{max} \leftarrow p \in P_{global} | \mathbb{E}(p) \geq \mathbb{E}(p'), \forall p' \in P_{global}$
        $P_{selected}.add(p^{max})$
        $P_{global}.remove(p^{max})$
        $\alpha \leftarrow$ *ESA*$(P_{selected})$
    end while
end if
return α

as follows. First, the graph $\mathcal{G}$, which corresponds to the NFVI topology is randomly generated. Second, the matching algorithm is applied to select a minimum number of links enabling the detection of up to a fixed number k of failures upper-bounded by the min degree d. Third, multiple samples of nodes states are simulated, and we compute the end-to-end Boolean path states. The number of simulated failures is varied to check the algorithm's behavior in different situations. Finally, the inference algorithm estimates the node states from end-to-end simulated measurements, and the estimations are compared to the real states to evaluate the accuracy of the monitoring system.

6.7.2 Results

Consider first the case of a network service chain A composed of 3 VNFs. Each network function has between 4 and 5 instances. The connections between the two consecutive sets of VNF instances are generated randomly. Each VNF instance has around 3 connections with the previous and the next sets of VNF instances. The matching algorithm selects a minimal number of connections enabling the detection of up to k simultaneous failed VMs, and k is fixed to 1 in these tests. Afterward, we simulate the node metrics as explained before. We vary the number

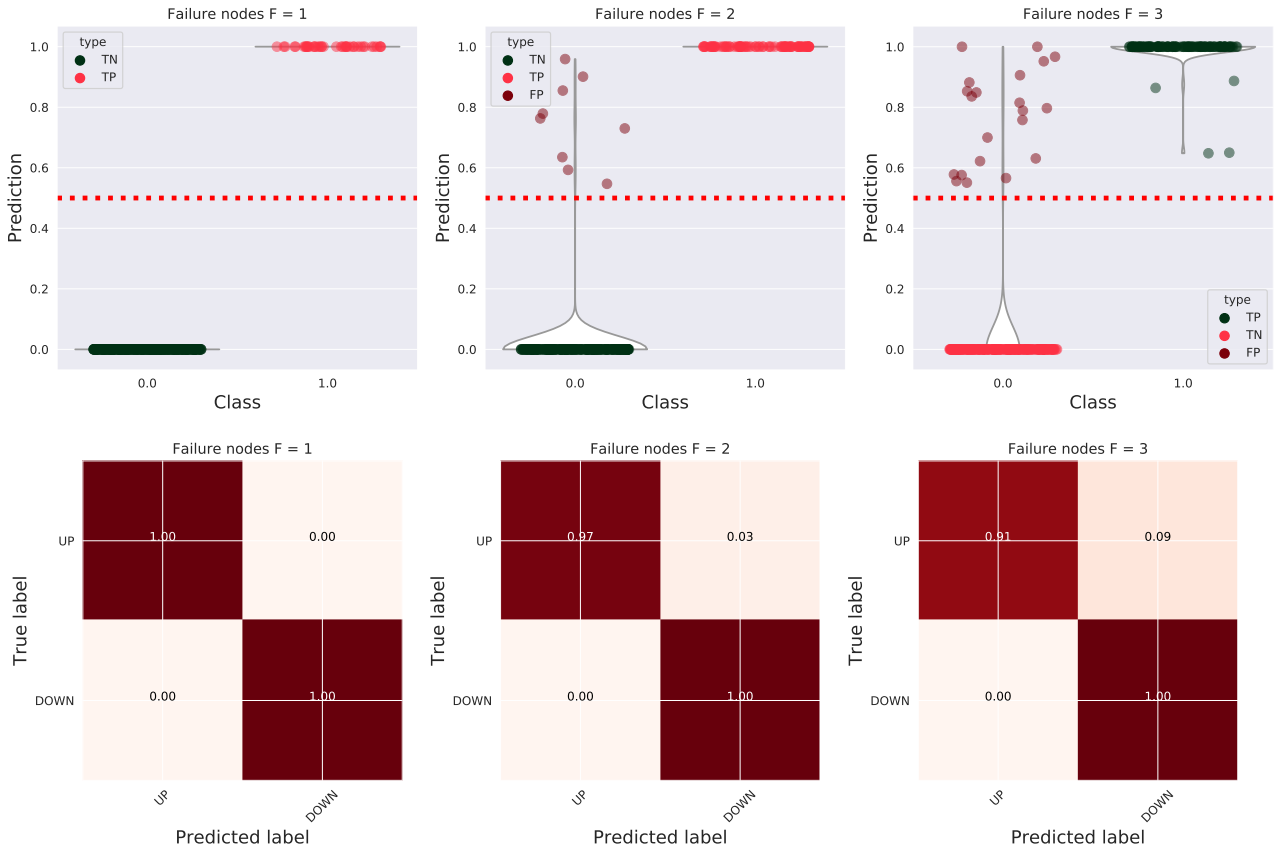


Figure 6.3 – Failure probability estimations and confusion matrix for 50 tests made with service chain A. The parameter k is fixed to 1, which means that the topology is designed to guarantee only the localization of only one failed node. In the tests, we vary the number of simultaneously failed nodes denoted, from 1 to 3.

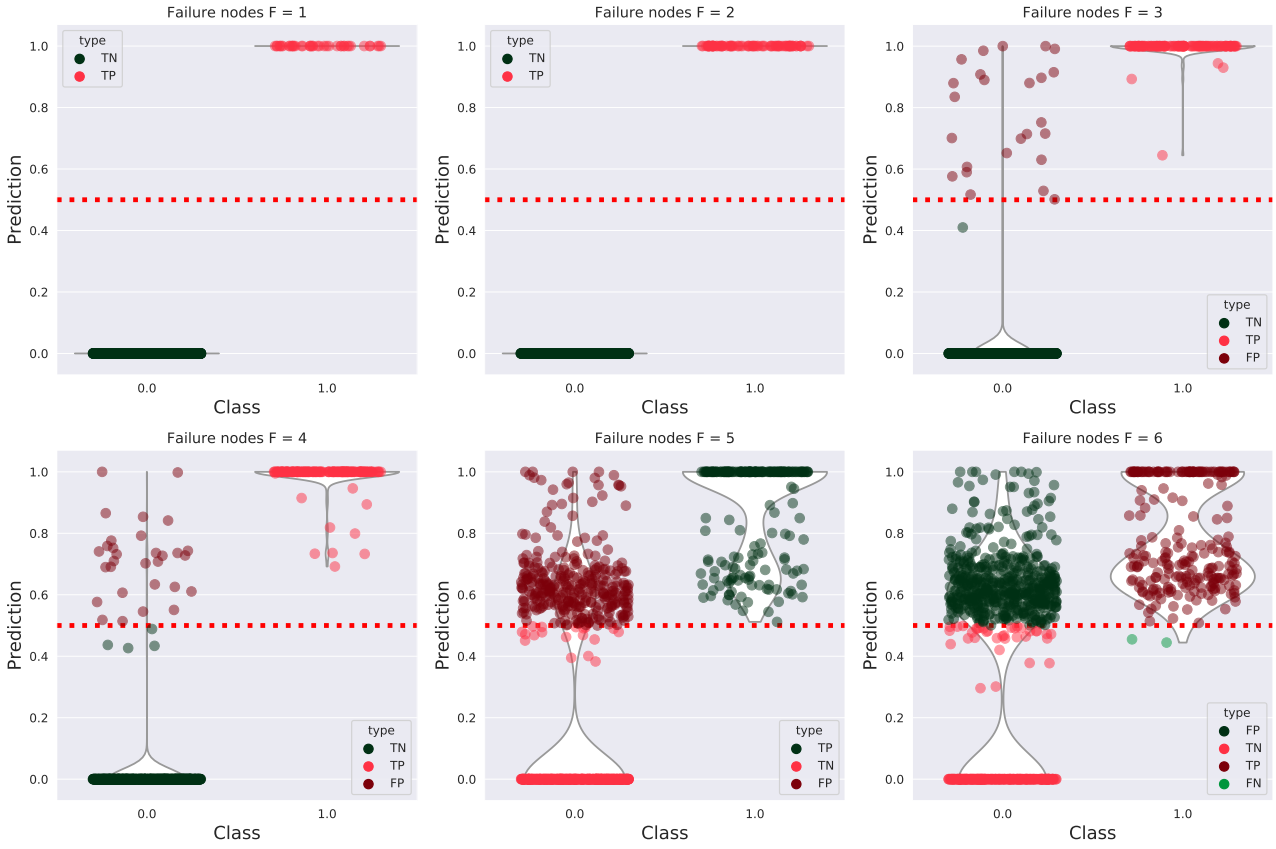


Figure 6.4 – Failure probability estimations and confusion matrix for 50 tests made with service chain B. The parameter k is equal to 1. We vary the number of failed nodes F from 1 to 6.

of generated node failures from 1 to 3.

Fig. 6.3 illustrates the probability dispersion and the confusion matrix for the node states made with 50 tests. With only one failure node, the monitoring system can localize them accurately in all the tests. These results are expected since the tested topology satisfies the sufficient condition for the detection of a single failed node. Therefore the number of false negatives and false positives is zero. However, this condition is not respected when more than one failure are generated as shown in the next tests, Fig. 6.3.

It is worth noting that if the sufficient condition for the detection of k failures is not satisfied, the monitoring system can still able to detect them if they occur. When the sufficient condition is not fulfilled, what we can say is that one or more situations (a combination of failed nodes) can exist where the monitoring system is unable to identify the failures in a deterministic way. The tests with 2 or 3 simultaneously failed nodes confirm this observation. In fact, with two and three failed nodes, the inference algorithm identifies all of them. However, the number of false positives increases a little bit to reach 3% and 9% respectively.

We make similar tests with a longer network service chain denoted by B, composed of 5 network functions. Fig. 6.4 illustrates the probability dispersion of the tests made with the second example. With only one failure, the monitoring algorithm identifies them accurately as in the first example. With two failures, the algorithm is still able to localize all malfunctions, which is not the case with the first example. This can be explained by the fact that the second topology is broader than the first one. Therefore, even if there are some cases where the monitoring system is unable to identify the unknown failure points, the probability of selecting these subsets of nodes during the simulations becomes extremely small due to the number of possible combinations that becomes very large. Then, starting from three failures, the numbers of false positives and false negatives start increasing.

6.8 Conclusion

In this chapter, we study the monitoring in network slicing and we focus on node anomaly localization. We provide a theoretical foundation to this problem and find necessary and sufficient conditions enabling the detection of up to k simultaneous failures in a network slice infrastructure for any given natural number k.

Afterward, these results allow us to establish a slice design strategy. The optimization process is formulated as a matching problem and solved with a classical max-flow algorithm. Finally, the end-to-end measurement performed on these topologies will be the input for the Boolean metrics inference algorithm to estimate the sate of each node and localize the potential failure points. The simulation results are consistent with our theoretical model presented earlier in the chapter.

CHAPTER 7

CONCLUSIONS OF THE THESIS

7.1 Introduction

5G networks will offer important capacities in terms of latency, bandwidth and massive connections. This evolution requires the use of new paradigms such as SDN and NFV, which offer flexibility, autonomy and dynamicity for the resource management and orchestration. To ensure a good network management, monitoring tools must give an accurate view of resource and traffic status. Meanwhile, several problems may be faced related to the access to the monitoring system and the availability of resources for information reporting.

The main thread of the thesis is the design of scalable monitoring solutions to avoid previous problems while keeping an acceptable accuracy. For this purpose, we studied the application of network tomography in this context. The main idea of this approach is to use end-to-end measurements to infer the state of the internal link or node metrics. There are several proposals in the literature, following this global idea, with variable performances, but they are not always applicable directly to our problem, for different reasons, including performance ones. In this work, we intend either to generalize their use to adapt it to the studied problem or to improve their performances.

7.2 Contributions of the thesis

The contributions of this thesis can be organized in four main parts:

— **Metrics inference algorithms:** In the first contribution we studied the inference of additive link metrics from unicast measurements. First, we extended a solution based on the Expectation-Maximization (EM) algorithm for tree topologies to general ones. However, this approach can be applied only with small graphs. Thus, we proposed the ESA technique which significantly enhances the performances and the scalability of previous algorithm. The general process is inspired by the steps of the EM procedure, but the elementary stages are based on the principles of genetic algorithms where we try to iteratively improve a solution population. Then, we proposed the Fast-ESA method which reduces further the computing time while keeping a good accuracy rate compared to

the ESA approach. Finally, we briefly discussed the application of the ESA method with non-additive metrics like the available bandwidth.

— **Neural network tomography:** The second contribution deals with the same inference problem but following a Machine Learning approach. We consider a classic Neural Network architecture of the Feed Forward type to learn the relation between link and path metrics. The model is trained with simulated data only, and it has good performances pretty close to those of our statistical method ESA. The solution requires a small volume of data which makes the training time very limited, compared to EM-based ideas. Indeed, it can be easily adapted according to the topology changes.

— **Monitors placement for probing cycles:** In the third contribution, we studied the monitors placement problem which is complementary to the inference task. The delay constraints in new networks are at the millisecond level. Monitoring systems must be highly accurate to make measurements at this scale. Indeed, in the probing techniques, the monitors must be highly synchronized which is not trivial in realistic conditions. Thus, we proposed to use only probing cycles for monitoring. The problem is modeled as a set covering task and we solve it using two methods, an optimal algorithm and and heuristic one. We prototyped and evaluated the solution over a Mininet emulator with a P4 implementation of source routing to build the probing cycles. In the thesis, we showed that the method has very good performances.

— **Node failures localization in an NFV infrastructure:** For the last contribution, we focused on failed nodes localization in NFV networks. We consider a Boolean metric where each node can have only two states, either up or down. Firstly, we give the necessary and sufficient conditions for the localization of k simultaneous failures in a virtual infrastructure. Then, based on these results, we proposed a strategy to design the network topology and the probing cycles that respects these conditions. The problem is formulated as a matching one and solved following a max-flow technique. The solution is evaluated with simulations and the results agree with the theoretical expectations.

7.3 Perspectives

Network tomography has been well studied in the literature, but we believe that there are new use cases and applications that can benefit from this discipline. Concerning our contributions we mention in this section some perspectives that can be considered.

— **Non-additive metrics:** In most of our contributions, we focused on the inference of additive metrics. This choice can be explained by the fact that perhaps the most significant metric, the delay, is additive, and other metrics such as the loss rate can can also be considered as additive. But our proposed solutions, both the ESA algorithm (and its en-

hancement F-ESA) or the Neural Network solution, can be applied to other metrics. This problem has been discussed briefly using the available bandwidth and Boolean metrics, but it requires more in-depth studies.

— **Topology learning:** The Machine Learning solution gives high accuracy with a pretty short computing time for either the learning or the inference steps. However, the slightest change in the network topology or paths can significantly degrade the results and this requires to repeat the training phase on the new one. So, it is interesting to improve the learning phase such that the model becomes more robust to these changes. This may require to extend the dataset used for learning, and perhaps to go for more advanced neural network architectures such as Graph Neural Networks.

— **Topology inference:** One of the most complicated problems in network tomography is topology inference.The use of Machine Learning can be an interesting research direction also for it, especially since the modeling of this problem is not obvious. For example, Neural Networks can provide generic models able to analyze measurements of varied nature and to capture correlations to infer the probable structure of the studied topology.

— **Mixing several metrics in one model:** Most of the solutions proposed in tomography are based on a single metric. The design of a model that takes as input several metrics of different nature is not obvious. Again, this problem can also be attacked using learning-based models. For this purpose, we need a rich and representative dataset in order to guarantee good results.

— **Programmable probing schemes:** The majority of tomography solutions are essentially based on two types of measurements, either unicast or multicast. Now, thanks to SDN technology, it is possible to design very sophisticated and more efficient probing schemes. In the thesis, we studied the use of loop paths. But thanks to the programmability of SDN networks, we can duplicate packets and redirect them more easily. We can add information fields in the headers and modify them, etc. Thus, the probing schemes can be adapted to the needed application to afford more useful information.

CONTRIBUTIONS OF THE THESIS

International conferences:

A. Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « Unicast Inference of Additive Metrics in General Network Topologies », *in*: *27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2019, Rennes, France, October 21-25, 2019*, IEEE Computer Society, 2019, pp. 107–115.

B. M. Rahali, J. Sanner, and G. Rubino, « TOM: a self-trained Tomography solution for Overlay networks Monitoring », *in*: *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, 2020, pp. 1–6.

C. M. Rahali, J. Sanner, and G. Rubino, « FEAL: A source routing Framework for Efficient Anomaly Localization », *in*: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

Submitted papers/journals:

D. Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « The Evolutionary Sampling Algorithm: a genetic algorithm approach for network tomography », *in*: .

E. Mohamed Rahali, Jean-Michel Sanner, Cao-Thanh PHAN, and Gerardo Rubino, « Failure nodes localization in NFV networks using network tomography », *in*: *2017 IEEE Conference on Computer Communications, INFOCOM 2021*, IEEE Computer Society, 2021.

Local workshops:

F. Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « Poster: SDN and Virtualisation: A new boost and opportunity for network tomography », *in*: SDN day 2018, Paris, 2018.

BIBLIOGRAPHY

[1] *ITU, Setting the Scene for 5G: Opportunities & Challenges*, tech. rep., 2018.

[2] Rui Castro et al., « Network Tomography: Recent Developments », *in*: *Statistical Science* 19 (Aug. 2004), DOI: 10.1214/088342304000000422.

[3] Earl Lawrence et al., « Network tomography: A review and recent developments », *in*: *In Fan and Koul, editors, Frontiers in Statistics*, College Press, 2006, pp. 345–364.

[4] A. Coates et al., « Internet tomography », *in*: *Signal Processing Magazine, IEEE* 19 (June 2002), pp. 47–65, DOI: 10.1109/79.998081.

[5] Nick G. Duffield et al., « Multicast-based loss inference with missing data », *in*: *IEEE J. Sel. Areas Commun.* 20.*4* (2002), pp. 700–713, DOI: 10.1109/JSAC.2002.1003037, URL: https://doi.org/10.1109/JSAC.2002.1003037.

[6] Mark Coates and Robert D. Nowak, « Network tomography for internal delay estimation », *in*: *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001, 7-11 May, 2001, Salt Palace Convention Center, Salt Lake City, Utah, USA, Proceedings*, IEEE, 2001, pp. 3409–3412, DOI: 10.1109/ICASSP.2001.940573, URL: https://doi.org/10.1109/ICASSP.2001.940573.

[7] Claudia Tebaldi and Mike West, « Bayesian Inference on Network Traffic Using Link Count Data », *in*: *Journal of the American Statistical Association* 93.*442* (1998), pp. 557–573, ISSN: 01621459, URL: http://www.jstor.org/stable/2670105.

[8] Ting He et al., « Fisher Information-based Experiment Design for Network Tomography », *in*: *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Portland, OR, USA, June 15-19, 2015*, ed. by Bill Lin et al., ACM, 2015, pp. 389–402, DOI: 10.1145/2745844.2745862, URL: https://doi.org/10.1145/2745844.2745862.

[9] Liang Ma et al., « Efficient Identification of Additive Link Metrics via Network Tomography », *in*: *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA*, IEEE Computer Society, 2013, pp. 581–590, DOI: 10.1109/ICDCS.2013.24, URL: https://doi.org/10.1109/ICDCS.2013.24.

[10] Liang Ma et al., « Node Failure Localization via Network Tomography », *in*: *Proceedings of the 2014 Internet Measurement Conference, IMC 2014, Vancouver, BC, Canada, November 5-7, 2014*, ed. by Carey Williamson, Aditya Akella, and Nina Taft, ACM, 2014, pp. 195–208, DOI: 10.1145/2663716.2663723, URL: https://doi.org/10.1145/2663716.2663723.

[11] Emanuele Goldoni and Marco Schivi, « End-to-End Available Bandwidth Estimation Tools, An Experimental Comparison », *in*: *Traffic Monitoring and Analysis*, ed. by Fabio Ricciato, Marco Mellia, and Ernst Biersack, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[12] *Scapy Python Library*, https://github.com/secdev/scapy/, 2019.

[13] Paula Branco, Luıés Torgo, and Rita P. Ribeiro, « A Survey of Predictive Modeling on Imbalanced Domains », *in*: *ACM Comput. Surv.* 49.*2* (Aug. 2016), 31:1–31:50, ISSN: 0360-0300.

[14] Takaya Saito and Marc Rehmsmeier, « The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets », *in*: *PloS one*, 2015.

[15] Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « Unicast Inference of Additive Metrics in General Network Topologies », *in*: *27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2019, Rennes, France, October 21-25, 2019*, IEEE Computer Society, 2019, pp. 107–115.

[16] M. Rahali, J. Sanner, and G. Rubino, « TOM: a self-trained Tomography solution for Overlay networks Monitoring », *in*: *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, 2020, pp. 1–6.

[17] M. Rahali, J. Sanner, and G. Rubino, « FEAL: A source routing Framework for Efficient Anomaly Localization », *in*: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[18] Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « The Evolutionary Sampling Algorithm: a genetic algorithm approach for network tomography », *in*:

[19] Mohamed Rahali et al., « Failure nodes localization in NFV networks using network tomography », *in*: *2017 IEEE Conference on Computer Communications, INFOCOM 2021*, IEEE Computer Society, 2021.

[20] Mohamed Rahali, Jean-Michel Sanner, and Gerardo Rubino, « Poster: SDN and Virtualisation: A new boost and opportunity for network tomography », *in*: SDN day 2018, Paris, 2018.

[21] Slawomir Kuklinski et al., « A reference architecture for network slicing », *in*: *4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018, Montreal, QC, Canada, June 25-29, 2018*, IEEE, 2018, pp. 217–221, DOI: 10.1109/NETSOFT.2018.8460057, URL: https://doi.org/10.1109/NETSOFT.2018.8460057.

[22] *NGMN, 5g white paper." [Online]. Available: https://www.ngmn.org/5g-white-paper.html*, tech. rep., 2016.

[23] Konstantinos Samdanis, Xavier Costa-Pérez, and Vincenzo Sciancalepore, « From network sharing to multi-tenancy: The 5G network slice broker », *in*: *IEEE Communications Magazine* 54.*7* (2016), pp. 32–39, DOI: 10.1109/MCOM.2016.7514161, URL: https://doi.org/10.1109/MCOM.2016.7514161.

[24] Muhammad Ali Imran and Ahmed Zoha, « Challenges in 5G: how to empower SON with big data for enabling 5G », *in*: *IEEE Network* 28.*6* (2014), pp. 27–33, DOI: 10.1109/MNET.2014.6963801, URL: https://doi.org/10.1109/MNET.2014.6963801.

[25] Nick McKeown et al., « OpenFlow: Enabling Innovation in Campus Networks », *in*: *SIGCOMM Comput. Commun. Rev.* 38.*2* (Mar. 2008), pp. 69–74, ISSN: 0146-4833, DOI: 10.1145/1355734.1355746, URL: https://doi.org/10.1145/1355734.1355746.

[26] Zohaib Latif et al., « A comprehensive survey of interface protocols for software defined networks », *in*: *J. Netw. Comput. Appl.* 156 (2020), p. 102563, DOI: 10.1016/j.jnca.2020.102563, URL: https://doi.org/10.1016/j.jnca.2020.102563.

[27] Xavier Costa-Pérez et al., « Radio access network virtualization for future mobile carrier networks », *in*: *IEEE Communications Magazine* 51.*7* (2013), pp. 27–35, DOI: 10.1109/MCOM.2013.6553675, URL: https://doi.org/10.1109/MCOM.2013.6553675.

[28] Salah-Eddine Elayoubi et al., « 5G RAN Slicing for Verticals: Enablers and Challenges », *in*: *IEEE Communications Magazine* 57.*1* (2019), pp. 28–34, DOI: 10.1109/MCOM.2018.1701319, URL: https://doi.org/10.1109/MCOM.2018.1701319.

[29] G. Wang et al., « Resource Allocation for Network Slices in 5G with Network Resource Pricing », *in*: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[30] Sihyung Lee, Kyriaki Levanti, and Hyong S. Kim, « Network monitoring: Present and future », *in*: *Comput. Networks* 65 (2014), pp. 84–98, DOI: 10.1016/j.comnet.2014.03.007, URL: https://doi.org/10.1016/j.comnet.2014.03.007.

[31] J. D. Case et al., *Simple Network Management Protocol (SNMP)*, RFC 3176, 1990.

[32] C. Estan, K. Keys, and D.and Varghese Moore, *Building a Better NetFlow: Technical Report*, tech. rep., Cooperative Association for Internet Data Analysis (CAIDA), June 2004.

[33] Sonia Panchen, Neil McKee, and Peter Phaal, *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*, RFC 3176, Sept. 2001.

[34] P. Tsai et al., « Network Monitoring in Software-Defined Networking: A Review », *in*: *IEEE Systems Journal* 12.*4* (Dec. 2018), pp. 3958–3969, ISSN: 2373-7816, DOI: 10.1109/JSYST.2018.2798060.

[35] Peng Sun et al., *HONE: Joint Host-Network Traffic Management in Software-Defined Networks*, 2014.

[36] Minlan Yu, Lavanya Jose, and Rui Miao, « Software Defined Traffic Measurement with OpenSketch », *in*: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, Lombard, IL: USENIX, 2013, pp. 29–42, ISBN: 978-1-931971-00-3, URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/yu.

[37] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, « OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks », *in*: *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8, DOI: 10.1109/NOMS.2014.6838228.

[38] S. R. Chowdhury et al., « PayLess: A low cost network monitoring framework for Software Defined Networks », *in*: *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9, DOI: 10.1109/NOMS.2014.6838227.

[39] J. Kempf et al., « Scalable fault management for OpenFlow », *in*: *2012 IEEE International Conference on Communications (ICC)*, June 2012, pp. 6606–6610.

[40] H. Xu and B. Li, « TinyFlow: Breaking elephants down into mice in data center networks », *in*: *2014 IEEE 20th International Workshop on Local Metropolitan Area Networks (LANMAN)*, May 2014, pp. 1–6, DOI: 10.1109/LANMAN.2014.7028620.

[41] B. Guan and S. Shen, « FlowSpy: An Efficient Network Monitoring Framework Using P4 in Software-Defined Networks », *in*: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Sept. 2019, pp. 1–5, DOI: 10.1109/VTCFall.2019.8891487.

[42] Giuseppe Aceto et al., « Cloud monitoring: A survey », *in*: *Comput. Networks* 57.*9* (2013), pp. 2093–2115, DOI: 10.1016/j.comnet.2013.04.001, URL: https://doi.org/10.1016/j.comnet.2013.04.001.

[43] Jonathan Stuart Ward and Adam Barker, « Observing the clouds: a survey and taxonomy of cloud monitoring », *in*: *J. Cloud Computing* 3 (2014), p. 24, DOI: 10.1186/s13677-014-0024-2, URL: https://doi.org/10.1186/s13677-014-0024-2.

[44] Kaniz Fatema et al., « A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives », *in*: *J. Parallel Distributed Comput.* 74.*10* (2014), pp. 2918–2933, DOI: 10.1016/j.jpdc.2014.06.007, URL: https://doi.org/10.1016/j.jpdc.2014.06.007.

[45] « Monitoring Cloud Computing by Layer, Part 1 », *in*: *IEEE Security Privacy* 9.*2* (2011), pp. 66–68.

[46] D. Verchere, « Cloud computing over telecom network », *in*: *2011 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference*, 2011, pp. 1–3.

[47] Yuh-Jye Chang et al., « Scalable and Elastic Telecommunication Services in the Cloud », *in*: *Bell Labs Tech. J.* 17.*2* (2012), pp. 81–96, DOI: 10.1002/bltj.21546, URL: https://doi.org/10.1002/bltj.21546.

[48] Alok Shrivastwa et al., *OpenStack: Building a Cloud Environment*, Packt Publishing, 2016, ISBN: 1787123189.

[49] Mathieu Lemay et al., *OpenDaylight Cookbook: Deploy and Operate Software-Defined Networking in Your Organization*, Packt Publishing, 2017, ISBN: 1786462303.

[50] Georgios Gardikis et al., « An integrating framework for efficient NFV monitoring », *in*: *IEEE NetSoft Conference and Workshops, NetSoft 2016, Seoul, South Korea, June 6-10, 2016*, IEEE, 2016, pp. 1–5, DOI: 10.1109/NETSOFT.2016.7502431, URL: https://doi.org/10.1109/NETSOFT.2016.7502431.

[51] L. Cao et al., « NFV-VITAL: A framework for characterizing the performance of virtual network functions », *in*: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 93–99.

[52] Steffen Becker et al., « The CloudScale Method for Managers », *in*: *Engineering Scalable, Elastic, and Cost-Efficient Cloud Computing Applications - The CloudScale Method*, ed. by Steffen Becker, Gunnar Brataas, and Sebastian Lehrig, Springer, 2017, pp. 149–165, DOI: 10.1007/978-3-319-54286-7_8, URL: https://doi.org/10.1007/978-3-319-54286-7%5C_8.

[53] Priyanka Naik, Dilip Kumar Shaw, and Mythili Vutukuru, « NFVPerf: Online performance monitoring and bottleneck detection for NFV », *in*: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, November 7-10, 2016*, IEEE, 2016, pp. 154–160, DOI: 10.1109/NFV-SDN.2016.7919491, URL: https://doi.org/10.1109/NFV-SDN.2016.7919491.

[54] Adam Pavlidis et al., « NFV-compliant Traffic Monitoring and Anomaly Detection based on Dispersed Vantage Points in Shared Network Infrastructures », *in*: *4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018, Montreal, QC, Canada, June 25-29, 2018*, IEEE, 2018, pp. 197–201, DOI: 10.1109/NETSOFT.2018.8460031, URL: https://doi.org/10.1109/NETSOFT.2018.8460031.

[55] Vincenzo Sciancalepore, Faqir Zarrar Yousaf, and Xavier Costa-Pérez, « z-TORCH: An Automated NFV Orchestration and Monitoring Solution », *in*: *IEEE Trans. Network and Service Management* 15.*4* (2018), pp. 1292–1306, DOI: 10.1109/TNSM.2018.2867827, URL: https://doi.org/10.1109/TNSM.2018.2867827.

[56] Carla Sauvanaud et al., « Anomaly Detection and Root Cause Localization in Virtual Network Functions », *in*: *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016*, IEEE Computer Society, 2016, pp. 196–206, DOI: 10.1109/ISSRE.2016.32, URL: https://doi.org/10.1109/ISSRE.2016.32.

[57] T. V. Landegem and R. Peschi, « Managing a connectionless virtual overlay network on top of an ATM network », *in*: *ICC 91 International Conference on Communications Conference Record*, 1991, 988–992 vol.2.

[58] Yan Chen, Chris Overton, and Randy H. Katz, « Internet Iso-bar: A Scalable Overlay Distance Monitoring System », English, *in*: *Journal of Computer Resource Management* (2002).

[59] Mehmet Demirci et al., « Multi-layer Monitoring of Overlay Networks », *in*: *Passive and Active Network Measurement, 10th International Conference, PAM 2009, Seoul, Korea, April 1-3, 2009. Proceedings*, ed. by Sue B. Moon, Renata Teixeira, and Steve Uhlig, vol. 5448, Lecture Notes in Computer Science, Springer, 2009, pp. 77–86, DOI: 10.1007/978-3-642-00975-4_8, URL: https://doi.org/10.1007/978-3-642-00975-4%5C_8.

[60] Chiping Tang and Philip K. McKinley, « A Distributed Approach to Topology-Aware Overlay Path Monitoring », *in*: *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, IEEE Computer Society, 2004, pp. 122–13, DOI: 10.1109/ICDCS.2004.1281575, URL: https://doi.org/10.1109/ICDCS.2004.1281575.

[61] Yongning Tang, Ehab Al-Shaer, and Bin Zhang, « Toward Globally Optimal Event Monitoring & Aggregation For Large-scale Overlay Networks », *in*: *Integrated Network Management, IM 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 21-25 May 2007*, IEEE, 2007, pp. 236–245, DOI: 10.1109/INM.2007.374788, URL: https://doi.org/10.1109/INM.2007.374788.

[62] Ramón Cáceres et al., « Multicast-based inference of network-internal loss characteristics », *in*: *IEEE Trans. Inf. Theory* 45.*7* (1999), pp. 2462–2480, DOI: 10.1109/18.796384, URL: https://doi.org/10.1109/18.796384.

[63] Liang Ma et al., « Identifiability of link metrics based on end-to-end path measurements », *in*: *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*, ed. by Konstantina Papagiannaki, P. Krishna Gummadi, and Craig Partridge, ACM, 2013, pp. 391–404, DOI: 10.1145/2504730.2504738, URL: https://doi.org/10.1145/2504730.2504738.

[64] Tian Bu et al., « Network tomography on general topologies », *in*: *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2002, June 15-19, 2002, Marina Del Rey, California, USA*, ed. by Richard R. Muntz, Margaret Martonosi, and Edmundo de Souza e Silva, ACM, 2002, pp. 21–30, DOI: 10.1145/511334.511338, URL: https://doi.org/10.1145/511334.511338.

[65] Francesco Lo Presti et al., « Multicast-based inference of network-internal delay distributions », *in*: *IEEE/ACM Trans. Netw.* 10.*6* (2002), pp. 761–775, DOI: 10.1109/TNET.2002.805026, URL: https://doi.org/10.1109/TNET.2002.805026.

[66] Nick G. Duffield et al., « Inferring Link Loss Using Striped Unicast Probes », *in*: *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, Anchorage, Alaska, USA, April 22-26, 2001*, IEEE Comptuer Society, 2001, pp. 915–923, DOI: 10.1109/INFCOM.2001.916283, URL: https://doi.org/10.1109/INFCOM.2001.916283.

[67] Y. Vardi, « Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data », *in*: *Journal of the American Statistical Association* 91.*433* (1996), pp. 365–377, ISSN: 01621459, URL: http://www.jstor.org/stable/2291416.

[68] Pedro Casas et al., « Efficient methods for traffic matrix modeling and on-line estimation in large-scale IP networks », *in*: *21st International Teletraffic Congress, ITC 2009, Paris, France, September 15-17, 2009*, IEEE, 2009, pp. 1–8, URL: http://ieeexplore.ieee.org/document/5300228/.

[69] Pedro Casas et al., « Volume Anomaly Detection in Data Networks: An Optimal Detection Algorithm vs. the PCA Approach », *in*: *Traffic Management and Traffic Engineering for the Future Internet, First Euro-NF Workshop, FITraMEn 2008, Porto, Portugal, December 11-12, Revised Selected Papers*, ed. by Rui Valadas and Paulo Salvador, vol. 5464, Lecture Notes in Computer Science, Springer, 2008, pp. 96–113, DOI: 10.1007/978-3-642-04576-9_7, URL: https://doi.org/10.1007/978-3-642-04576-9%5C_7.

[70] William I. Notz, « Optimum Experimental Designs (A. C. Atkinson and A. N. Donev) », *in*: *SIAM Review* 36.*2* (1994), pp. 315–316, DOI: 10.1137/1036083, URL: https://doi.org/10.1137/1036083.

[71] Harsh Singhal and George Michailidis, « Optimal Sampling in State Space Models with Applications to Network Monitoring », *in*: vol. 36, June 2008, pp. 145–156, DOI: 10.1145/1384529.1375474.

[72] Liang Ma et al., « Monitor placement for maximal identifiability in network tomography », *in*: *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, IEEE, 2014, pp. 1447–1455, DOI: 10.1109/INFOCOM.2014.6848079, URL: https://doi.org/10.1109/INFOCOM.2014.6848079.

[73] Ting He et al., « Robust and Efficient Monitor Placement for Network Tomography in Dynamic Networks », *in*: *IEEE/ACM Trans. Netw.* 25.*3* (2017), pp. 1732–1745, DOI: 10.1109/TNET.2016.2642185, URL: https://doi.org/10.1109/TNET.2016.2642185.

[74] Abishek Gopalan and Srinivasan Ramasubramanian, « On Identifying Additive Link Metrics Using Linearly Independent Cycles and Paths », *in*: *IEEE/ACM Trans. Netw.* 20.*3* (2012), pp. 906–916, DOI: 10.1109/TNET.2011.2174648, URL: https://doi.org/10.1109/TNET.2011.2174648.

[75] Abishek Gopalan and Srinivasan Ramasubramanian, « On the Maximum Number of Linearly Independent Cycles and Paths in a Network », *in*: *IEEE/ACM Trans. Netw.* 22.*5* (2014), pp. 1373–1388, DOI: 10.1109/TNET.2013.2291208, URL: https://doi.org/10.1109/TNET.2013.2291208.

[76] Rongwei Yang et al., « On the Optimal Monitor Placement for Inferring Additive Metrics of Interested Paths », *in*: *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, IEEE, 2018, pp. 2141–2149, DOI: 10.1109/INFOCOM.2018.8486423, URL: https://doi.org/10.1109/INFOCOM.2018.8486423.

[77] D. Z. Du and Frank K. Hwang, « Combinatorial Group Testing and Its Applications », *in*: 2000.

[78] Robert Dorfman, « The Detection of Defective Members of Large Populations », *in*: *Annals of Mathematical Statistics* 14 (1943), pp. 436–440.

[79] Mahdi Cheraghchi et al., « Graph-Constrained Group Testing », *in*: *IEEE Trans. Inf. Theory* 58.*1* (2012), pp. 248–262, DOI: 10.1109/TIT.2011.2169535, URL: https://doi.org/10.1109/TIT.2011.2169535.

[80] Liang Ma et al., « Network Capability in Localizing Node Failures via End-to-End Path Measurements », *in*: *IEEE/ACM Trans. Netw.* 25.*1* (2017), pp. 434–450, DOI: 10.1109/TNET.2016.2584544, URL: https://doi.org/10.1109/TNET.2016.2584544.

[81] Novella Bartolini et al., « On Fundamental Bounds of Failure Identifiability by Boolean Network Tomography », *in*: *CoRR* abs/1903.10636 (2019), arXiv: 1903.10636, URL: http://arxiv.org/abs/1903.10636.

[82] Nicola Galesi and Fariba Ranjbar, « Tight Bounds for Maximal Identifiability of Failure Nodes in Boolean Network Tomography », *in*: *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, IEEE Computer Society, 2018, pp. 212–222, DOI: 10.1109/ICDCS.2018.00030, URL: https://doi.org/10.1109/ICDCS.2018.00030.

[83] Satyajeet Ahuja, Srinivasan Ramasubramanian, and Marwan Krunz, « SRLG Failure Localization in All-Optical Networks Using Monitoring Cycles and Paths », *in*: *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, IEEE, 2008, pp. 700–708, DOI: 10.1109/INFOCOM.2008.45, URL: https://doi.org/10.1109/INFOCOM.2008.45.

[84] Sylvia Ratnasamy and Steven McCanne, « Inference of Multicast Routing Trees and Bottleneck Bandwidths Using End-to-end Measurements », *in*: *Proceedings IEEE INFOCOM '99, The Conference on Computer Communications, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, The Future Is Now, New York, NY, USA, March 21-25, 1999*, IEEE Computer Society, 1999, pp. 353–360, DOI: 10.1109/INFCOM.1999.749302, URL: https://doi.org/10.1109/INFCOM.1999.749302.

[85] Jian Ni et al., « Efficient and dynamic routing topology inference from end-to-end measurements », *in*: *IEEE/ACM Trans. Netw.* 18.*1* (2010), pp. 123–135, DOI: 10.1145/1816288.1816298, URL: http://doi.acm.org/10.1145/1816288.1816298.

[86] Mark Coates et al., « Maximum likelihood network topology identification from edge-based unicast measurements », *in*: *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2002, June 15-19, 2002, Marina Del Rey, California, USA*, ed. by Richard R. Muntz, Margaret Martonosi, and Edmundo de Souza e Silva, ACM, 2002, pp. 11–20, DOI: 10.1145/511334.511337, URL: https://doi.org/10.1145/511334.511337.

[87] Anirudh Sabnis, Ramesh K. Sitaraman, and Donald F. Towsley, « OCCAM: An Optimization Based Approach to Network Inference », *in*: *SIGMETRICS Perform. Evaluation Rev.*

46.*2* (2018), pp. 36–38, DOI: 10.1145/3305218.3305232, URL: https://doi.org/10.1145/3305218.3305232.

[88] Yilei Lin et al., « Looking Glass of NFV: Inferring the Structure and State of NFV Network from External Observations », *in*: *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*, IEEE, 2019, pp. 1774–1782, DOI: 10.1109/INFOCOM.2019.8737393, URL: https://doi.org/10.1109/INFOCOM.2019.8737393.

[89] C. Liu et al., « Multicast vs. unicast for loss tomography on tree topologies », *in*: *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, pp. 312–317.

[90] Earl Lawrence, George Michailidis, and Vijayan N. Nair, « Network Delay Tomography Using Flexicast Experiments », *in*: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 68.*5* (2006), pp. 785–813.

[91] Ramón Cáceres et al., « Multicast-based inference of network-internal loss characteristics », *in*: *IEEE Trans. Information Theory* 45.*7* (1999), pp. 2462–2480.

[92] W. Zhu, « Loss Tomography in General Topology », *in*: *arXiv e-prints* 1603.07825 (2016).

[93] Ke Deng et al., « Fast Parameter Estimation in Loss Tomography for Networks of General Topology », *in*: *The Annals of Applied Statistics* 10 (Oct. 2015), DOI: 10.1214/15-AOAS883.

[94] V. N. Padmanabhan, L. Qiu, and H. J. Wang, « Server-based inference of Internet link lossiness », *in*: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 1, 2003, 145–155 vol.1.

[95] Ke Deng et al., « On Delay Tomography: Fast Algorithms and Spatially Dependent Models », *in*: *IEEE Trans. Signal Processing* 60.*11* (2012), pp. 5685–5697, DOI: 10.1109/TSP.2012.2210712.

[96] Nick Duffield, « Simple Network Performance Tomography », *in*: *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, Miami Beach, FL, USA, 2003, pp. 210–215, ISBN: 1-58113-773-7.

[97] Yan Qiao, Xuesong Qiu, Luoming Meng and Ran Gu, « Efficient Loss Inference Algorithm Using Unicast End-to-End Measurements », *in*: *Network System Management* 21 (2013), pp. 169–193.

[98] N. G. Duffield et al., « Multicast topology inference from measured end-to-end loss », *in*: *IEEE Transactions on Information Theory* 48.*1* (2002), pp. 26–45.

[99] Albert Mestres et al., « Knowledge-Defined Networking », *in*: *Computer Communication Review* 47 (2017), pp. 2–10.

[100] Simon Knight et al., « The Internet Topology Zoo », *in*: *IEEE Journal on Selected Areas in Communications* 29.*9* (2011), pp. 1765–1775.

[101] S. Orlowski et al., « SNDlib 1.0&Mdash;Survivable Network Design Library », *in*: *Netw.* 55.*3* (May 2010), pp. 276–286, ISSN: 0028-3045, DOI: 10.1002/net.v55:3.

[102] Liang Ma, Ziyao Zhang, and Mudhakar Srivatsa, « Neural Network Tomography », *in*: *CoRR* abs/2001.02942 (2020), arXiv: 2001.02942, URL: http://arxiv.org/abs/2001.02942.

[103] Diederik P. Kingma and Jimmy Ba, « Adam: A Method for Stochastic Optimization », *in*: *CoRR* abs/1412.6980 (2015).

[104] Yan Chen, David Bindel, and Randy H. Katz, « Tomography-based Overlay Network Monitoring », *in*: *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, Miami Beach, FL, USA: ACM, 2003, ISBN: 1-58113-773-7.

[105] G. H. Golub and C. Reinsch, « Singular Value Decomposition and Least Squares Solutions », *in*: *Handbook for Automatic Computation: Volume II: Linear Algebra*, ed. by F. L. Bauer et al., Berlin, Heidelberg: Springer Berlin Heidelberg, 1971, pp. 134–151.

[106] Cristel Pelsser et al., « From Paris to Tokyo: on the suitability of ping to measure latency », *in*: *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*, 2013, pp. 427–432.

[107] Clarence Filsfils et al., « The Segment Routing Architecture », *in*: *2015 IEEE Global Communications Conference, GLOBECOM 2015, San Diego, CA, USA, December 6-10, 2015*, 2015, pp. 1–6.

[108] Francois Aubry et al., « SCMon: Leveraging segment routing to improve network monitoring », *in*: *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, 2016, pp. 1–9.

[109] Xiaoqian Li and Kwan L. Yeung, « ILP Formulation for Monitoring-Cycle Construction Using Segment Routing », *in*: *43rd IEEE Conference on Local Computer Networks, LCN 2018, Chicago, IL, USA, October 1-4, 2018*, 2018, pp. 485–492.

[110] Zahraa N. Abdullah, Imtiaz Ahmad, and Iftekhar Hussain, « Segment Routing in Software Defined Networks: A Survey », *in*: *IEEE Communications Surveys and Tutorials* 21.*1* (2019), pp. 464–486.

[111] Ahmed Bashandy et al., *Segment Routing with MPLS data plane*, Internet-Draft draft-ietf-spring-segment-routing-mpls-22, Work in Progress, Internet Engineering Task Force, May 2019, 38 pp.

[112] Eduardo Moreno, Alejandra Beghelli, and Filippo Cugini, « Traffic engineering in segment routing networks », *in*: *Computer Networks* 114 (2017), pp. 23–31.

[113] Xiaoqian Li and Kwan L. Yeung, « Designing Network Monitoring Schemes Based on Segment Routing », *in*: *2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017*, 2017, pp. 1–6.

[114] A. Dusia and A. S. Sethi, « Recent Advances in Fault Localization in Computer Networks », *in*: *IEEE Communications Surveys Tutorials* 18.*4* (Fourthquarter 2016), pp. 3030–3051, ISSN: 1553-877X, DOI: 10.1109/COMST.2016.2570599.

[115] Malgorzata Steinder and Adarshpal S. Sethi, « A survey of fault localization techniques in computer networks », *in*: *Science of Computer Programming* 53 (Nov. 2004), pp. 165–194, DOI: 10.1016/j.scico.2004.01.010.

[116] Abishek Gopalan and Srinivasan Ramasubramanian, « On Identifying Additive Link Metrics Using Linearly Independent Cycles and Paths », *in*: *IEEE/ACM Transactions on Networking - TON* 20 (June 2012), pp. 906–916, DOI: 10.1109/TNET.2011.2174648.

[117] T. He et al., « Robust and Efficient Monitor Placement for Network Tomography in Dynamic Networks », *in*: *IEEE/ACM Transactions on Networking* 25.*3* (June 2017), pp. 1732–1745.

[118] W. Dong et al., « Optimal Monitor Assignment for Preferential Link Tomography in Communication Networks », *in*: *IEEE/ACM Transactions on Networking* 25.*1* (Feb. 2017), pp. 210–223.

[119] L. Ma et al., « Link identifiability in communication networks with two monitors », *in*: *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2013, pp. 1513–1518.

[120] Alberto Caprara, Paolo Toth, and Matteo Fischetti, « Algorithms for the Set Covering Problem », *in*: *Annals of Operations Research* 98.*1* (Dec. 2000), pp. 353–371.

[121] Pat Bosshart et al., « P4: Programming Protocol-independent Packet Processors », *in*: *SIGCOMM Comput. Commun. Rev.* 44.*3* (July 2014), pp. 87–95, ISSN: 0146-4833, DOI: 10.1145/2656877.2656890, URL: http://doi.acm.org/10.1145/2656877.2656890.

[122] *Behavioral-model*, https://github.com/p4lang/behavioral-model, 2019.

[123] *Source Routing implementation*, https://github.com/p4lang/tutorials/tree/master/exercises/source_routing, 2019.

[124] Roshan Kumari and Saurabh Kr. Srivastava, « Machine Learning: A Review on Binary Classification », *in*: 2017.

[125] Carla Sauvanaud et al., « Anomaly Detection and Root Cause Localization in Virtual Network Functions », *in*: *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016*, IEEE Computer Society, 2016, pp. 196–206, DOI: 10.1109/ISSRE.2016.32, URL: https://doi.org/10.1109/ISSRE.2016.32.

[126] Michail-Alexandros Kourtis et al., « Statistical-based anomaly detection for NFV services », *in*: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, November 7-10, 2016*, IEEE, 2016, pp. 161–166, DOI: 10.1109/NFV-SDN.2016.7919492, URL: https://doi.org/10.1109/NFV-SDN.2016.7919492.

[127] Emna Salhi, Samer Lahoud, and Bernard Cousin, « Heuristics for Joint Optimization of Monitor Location and Network Anomaly Detection », *in*: *Proceedings of IEEE International Conference on Communications, ICC 2011, Kyoto, Japan, 5-9 June, 2011*, IEEE, 2011, pp. 1–5, DOI: 10.1109/icc.2011.5962551, URL: https://doi.org/10.1109/icc.2011.5962551.

[128] Emna Salhi, Samer Lahoud, and Bernard Cousin, « Joint optimization of monitor location and network anomaly detection », *in*: *The 35th Annual IEEE Conference on Local Computer Networks, LCN 2010, 10-14 October 2010, Denver, Colorado, USA, Proceedings*, IEEE Computer Society, 2010, pp. 204–207, DOI: 10.1109/LCN.2010.5735702, URL: https://doi.org/10.1109/LCN.2010.5735702.

[129] Liang Ma et al., « Node Failure Localization via Network Tomography », *in*: *Proceedings of the 2014 Internet Measurement Conference, IMC 2014, Vancouver, BC, Canada, November 5-7, 2014*, ed. by Carey Williamson, Aditya Akella, and Nina Taft, ACM, 2014, pp. 195–208, DOI: 10.1145/2663716.2663723, URL: https://doi.org/10.1145/2663716.2663723.

[130] Novella Bartolini, Ting He, and Hana Khamfroush, « Fundamental limits of failure identifiability by boolean network tomography », *in*: *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, IEEE, 2017, pp. 1–9, DOI: 10.1109/INFOCOM.2017.8057091, URL: https://doi.org/10.1109/INFOCOM.2017.8057091.

[131] Nicola Galesi and Fariba Ranjbar, « Tight Bounds for Maximal Identifiability of Failure Nodes in Boolean Network Tomography », *in*: *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, IEEE Computer Society, 2018, pp. 212–222, DOI: 10.1109/ICDCS.2018.00030, URL: https://doi.org/10.1109/ICDCS.2018.00030.

[132] N. N. Jara, G. Rubino, and R. Vallejos, « A Method for Joint Routing, Wavelength Dimensioning and Fault Tolerance for any set of simultaneous failures on Dynamic WDM Optical Networks », *in*: *Optical Fiber Technology* 38 (2017), pp. 30–40.

[133] John E. Hopcroft and Richard M. Karp, « An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs », *in*: *SIAM Journal on Computing* 2.4 (1973), pp. 225–231, DOI:

10.1137/0202019, eprint: https://doi.org/10.1137/0202019, URL: https://doi.org/10.1137/0202019.

[134] Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett, « Group Testing: An Information Theory Perspective », *in*: *Found. Trends Commun. Inf. Theory* 15.*3-4* (2019), pp. 196–392.

ACRONYMS

3GPP 3rd Generation Partnership Project.

CAPEX Capital Expense.

CN Core Network.

EM Expectation-Maximization.

ESA Evolutionary Sampling Algorithm.

ETSI European Telecommunications Standards Institute.

ILP Integer Linear Programming.

KPIs Key Performance Indicators.

LTE Long Term Evolution.

MANO Management and Orchestration.

MIB Management Information Base.

MLE Maximum Likelihood Estimator.

NFV Network Functions Virtualisation.

NFVI Network Functions Virtualisation Infrastructure.

NGMN Next Generation Mobile Networks.

OF OpenFlow.

ONF Open Networking Foundation.

OPEX Operational Expense.

OVS Open vSwitch.

P4 Programming Protocol-independent Packet Processors.

QoS Quality of Service.

RAN Radio Access Network.

SDN Software Defined Network.

SLA Service Level Agreement.

SNMP Simple Network Management Protocol.

SON Self-Organizing Networks.

VIM Virtualized Infrastructure Manager.

VNF Virtual Network Function.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Les réseaux SDN et NFV : Un nouvel élan et une nouvelle opportunité pour la tomographie en réseaux

Mot clés : SDN, NFV, Surveillance des réseaux, Tomographie des réseaux, Détection d'anomalies

Résumé : Le concept de découpage du réseau ("slicing") promet une flexibilité et une autonomie importantes pour la gestion des ressources. Grâce à ses principales caractéristiques, qui s'appuient essentiellement sur les technologies NFV et SDN, les nouveaux services de communication peuvent être conçus et déployés beaucoup plus rapidement qu'auparavant. Cependant, le maintien du niveau nécessaire de fiabilité reste un problème majeur. L'une de ses conséquences est que la surveillance de l'infrastructure réseau dédiée à cette classe de services est un défi essentiel, défi que nous abordons dans cette thèse via l'utilisation des techniques de tomographie.

La tomographie étudie l'inférence des performances du réseau interne à partir de mesures externes. Nous étudions, entre autre, son application dans le contexte du slicing et des réseaux virtuels. Tout d'abord, nous présentons l'algorithme d'échantillonnage évolutif (ESA) comme un outil permettant d'inférer des métriques de nœuds ou de liens (dites "cachés", c'est-à-dire, intérieures) à partir des mesures de bout en bout, en suivant une approche de type algorithme génétique. Il améliore la précision et le temps de calcul des solutions existantes basées sur l'algorithme espérance-maximisation (EM). Ensuite, nous utilisons une architecture de réseau neuronal entraîné comme outil d'inférence pour traiter le même problème. Le modèle apprend avec un petit volume de données simulées ce qui permet de le répéter rapidement en fonction des mises à jour du réseau. Deuxièmement, nous avons abordé le problème de placement des moniteurs. Nous avons étudié un cas particulier où seulement des cycles sont utilisés. La motivation de ce choix est d'éviter les contraintes liées à la synchronisation entre les nœuds. La programmabilité du réseau offerte par le SDN permet le déploiement de ce type de chemin personnalisé. Cette problématique est formulée comme un problème de couverture d'ensemble et nous avons proposé deux approches pour le résoudre, un algorithme optimal et un autre heuristique. Enfin, nous avons étudié le problème de la détection d'anomalies dans les réseaux NFV. Nous appliquons les principes de la tomographie booléenne, où chaque nœud ne peut avoir que deux états : opérationnel ou défaillant. Nous donnons des conditions nécessaires et suffisantes sur la topologie, plus une stratégie de sondage qui garantissent la localisation d'une limite maximale fixe de nœuds simultanément défaillants.

Title: SDN and NFV networks: A new boost and opportunity for network tomography

Keywords: SDN, NFV, Network monitoring, Network Tomography, Anomaly Detection

Abstract: The network slicing concept promises significant flexibility and autonomy for network management. Thanks to its main key features, heavily relying on the NFV and the SDN technologies, new communication services can be designed and deployed much faster than before. However, maintaining the necessary reliability level of conventional networks remains a major open problem. One of its consequences is that the monitoring of the network infrastructure dedicated to this class of services is an essential challenge, which we address in this thesis through tomography methods.

Internet tomography studies the inference of the internal network performances from end-to-end measurements. We study, among other things, its application in the context of network slicing and virtual networks. First, we present the Evolutionary Sampling Algorithm (ESA) as a tool for the inference of hidden nodes' or links' metrics from end-to-end measurements, following a genetic algorithm approach. It enhances the accuracy and the computing time of existing solutions based on the Expectation Maximization (EM) algorithm. Then, we use a trained neural network architecture as an inference tool to deal with the same problem. The model is trained with only a few simulated data. The training time is very short which gives the possibility to repeat it according to the network updates. Second, we addressed the monitors' placement problem. We have studied a particular case where only cycle paths are used. The motivation behind this choice is to avoid the constraints related to synchronization between nodes. The network programmability offered by SDN allows the deployment of such customized type of probing paths. This task is formulated as a set covering problem and we proposed two approaches to solve it, an optimal algorithm and a heuristic one. Finally, we studied the anomaly detection problem in NFV networks. We apply the principles of Boolean network tomography, where we consider that each node can have only two states: up or down. We give necessary and sufficient conditions about the network topology plus a strategy for building the probing paths that guarantee the localization of a fixed maximum number of simultaneously failed nodes.